spectrum adventures

a guide to playing and writing adventures

tony bridge & roy carnell

spectrum adventures

a guide to playing and writing adventures

tony bridge & roy carnell

First published 1983 by:
Sunshine Books (an imprint of Scot Press Ltd.)
12–13 Little Newport Street,
London WC2R 3LD

Reprinted 1984

Copyright © Tony Bridge and Roy Carnell

™ ZX and ZX Spectrum are Trade Marks of Sinclair Research Ltd.

ISBN 0 946408 07 6

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording and/or otherwise, without the prior written permission of the Publishers.

Cover design by Graphic Design Ltd.
Illustration by Stuart Hughes.
Typeset and printed in England by Commercial Colour Press, London E7.

FOR JILLY — My real life adventure

Thanks to Mary and Stuart Galloway, for their patience — and to Magra, for waiting!

Contents

		Page
Part 1	The Spectrum — your hero	
1	Origins	11
2	The First Software	17
3	The Hobbit	27
4	Graphic Adventures	31
5	Meet the Cast	43
Part 2	The Eye of the Star Warrior	
6	Why a Graphic Adventure?	61
7	The Legend	63
8	Create your own Dungeon!	65
9	All that Glisters...	75
10	Graphic Violence	79
11	Make your Move	85
12	Monster! Monster!	93
13	Trusty Weapons	99
14	I Spell a Rat...	117
15	Nice and Tidy...	123
Endplay		131
Appendices		
A	Defined Graphics	133
B	Instructions	141
C	The Complete Listing	145
	Table of Variables	

Contents in detail

Part 1

CHAPTER 1
Origins
Wargames and Dungeons and Dragons — the origins of adventure on computers.

CHAPTER 2
The First Software
The early history of Sinclair computers and the first adventure software. A blow-by-blow account of a game of adventure on the Spectrum.

CHAPTER 3
The Hobbit

CHAPTER 4
Graphic Adventures
Sorcerer's Castle, The Valley, Halls of the Things.

CHAPTER 5
Meet the Cast
The treasures, monsters and situations met in a typical adventure.

Part 2

CHAPTER 6
Why a Graphic Adventure?

CHAPTER 7
The Legend of the Star Warrior

CHAPTER 8
Create your own Dungeon...

CHAPTER 9
All that Glisters...
Put some treasure and lots of monsters in it...

CHAPTER 10
Graphic Violence
. . . and some graphics. . .

CHAPTER 11
Make your Move
. . . and then start moving around.

CHAPTER 12
Monster! Monster
But the monsters will come to get you.

CHAPTER 13
Trusty Weapons
How to bash a monster (or be bashed).

CHAPTER 14
I spell a Rat
Spells and things.

CHAPTER 15
Nice and Tidy. . .
Tidying up — Healing Wells and Wizard's Hands, and several useful routines for your own adventure.

APPENDIX A
The graphics used in the program.

APPENDIX B
Instructions for playing *The Eye of the Star Warrior*.

APPENDIX C
The complete listing of the game, with the variables used.

PART 1

Your Spectrum,
—
Your Hero

CHAPTER 1
Origins

Switch the light on, quick! There, that's better, now we can see a little further ahead into the gloom. The strange noises went away when the light went on, and now all is quiet. Ahead, an opening in the cave wall — let's go through and see what lies ahead for us.

Aha! What's this? A Black Rod lying on the damp floor of the cave. Go on, pick it up. Anything else around? No, that's it, so let's go further into the catacombs. In the next cave, an empty Whicker Cage lies discarded in one corner. Is it a trap? Try poking it with the Rod — wave the Rod at it — nothing happens, so we might as well pick it up.

Now the faint sound of a bird singing comes from the next cave. Quietly moving into the cave, we can see a Bird happily singing on a rock not far away. That must be what the Whicker Cage is for! Right, let's catch the little blighter. But it flies away — something is frightening it. Well, the Black Rod looks a bit menacing, so let's drop it. That's better, the Bird has settled again, and is merrily singing, oblivious to us creeping towards it with our Cage. The bird is caught!

On again, pausing to pick up the Rod (no doubt it will come in handy later), to the next cave, with Lamp held high before us. Suddenly a vast shadow rears up before us from the gloom! A huge Green Snake fixes us with its eye and sways before us. There is no other exit apparent, so we must get past this beast in order to progress on our quest. Let's stop and think for a moment.

Waving the Rod might work this time — but no, the Snake isn't worried by that! How about using the Rod as a polevault? The Snake is too big to vault over. The little Bird is still singing away merrily — now, wait a moment here, maybe we can feed the Bird to the Snake. After all, the Snake is probably hungry, and this might keep it busy long enough for us to rush past and away. So let's free the bird and see what happens. Now would you believe it? The Snake, frightened by the fluttering of wings, hisses violently and slithers away into the blackness, leaving us free to continue our quest.

This is a typical scenario from a computer-assisted adventure, in which overt violence does not play a large part, but there are variations in which your sword-arm and fighting abilities take precedence over your reasoning powers. We'll look at some of these variations later, but for now, let's examine what we mean by adventure.

To find the origins of computer-assisted adventure, we have to look first at another pastime from the pre-home computer age. (Remember those days, before you spent your evenings glued to the latest version of Space Invaders, or Zaxxon?)

A pastime as old as chess itself, and nearly as old as another pastime!

War-games have been fought since chieftains had more than half-a-dozen men to their armies — after all, even the most basic of fighting manoeuvres need practising, and what better way of doing that than making a game of it, thus giving some incentive to the proceedings?

Some very complex versions of war-games have been mounted, but the one that most people will recognise as such is the tabletop war. This has been a popular form of recreation and a military training technique since the 17th century.

During the 1960s, however, books of rules to use with war-games made a commercial appearance, and became very popular. These rulebooks covered the smallest details of the period concerned, such as uniforms, weapons and logistics. The main periods were ancient, medieval, Napoleonic and modern.

There were, within these categories, many sub-categories, and one of these was medieval fantasy. Dave Arneson, of the Castle and Crusade Society in the United States, began a vast campaign, and expanded the original rules to offer a complete environment for the players. These rules evolved, with the help of Gary Gygax, into one of the most successful games of the century — *Dungeons & Dragons*, published by Tactical Studies Rules. *Dungeons & Dragons*, from TSR, presents the player with a highly stylised system of play, where nothing is left to chance, but everything played according to tables. It became extremely popular, and is still to this day. War-gaming in general now went in two directions — one headed by the traditionalist, moving his armies of miniature lead soldiers about his tabletop, measuring explosion effects with a steel rule, and the other by a generation weaned on board games.

Their preferred playing pieces were small cardboard counters, representing anything from individual soldiers, planes or ships, through battalions and divisions to whole navies and armies. These war-games are played on a board, marked across by hexagons to facilitate movement, and various representations of landscapes. The box games range from individual combat to global warfare, and from utter simplicity to staggering complexity, needing several days (or weeks?) to play.

However, Gygax and Arneson, in the early 1970s, forged a path of their own through the uncharted world of fantasy war-gaming and created a unique system of playing, with their game of *Dungeons & Dragons*. Although the name is jealously guarded by TSR, the holders of the copyright, D&D is often used as a generic term for this type of combat system.

The official term, however, for all the games that have sprung from *D &D*, is Role-Playing Games, or RPGs for short.

Let's have a brief look at how these RPGs are played.

A complex of, originally, Dungeons (though this might nowadays be anything from a spaceship to post-nuclear metropolis) is designed before play starts by the Dungeon-Master. The DM then invites other players to explore his creation. It's possible for just one player to undertake this task, but a party of two or more players is preferable, as they can bounce ideas off each other, thus making for a more enjoyable game.

Each player controls a character, often represented by a little model (shades of a, by now, fairly distant relation!), and the party moves forward together. The play is made as real as possible, and the participants, apart from the DM, have no idea what awaits them as they venture forth. As they get to a closed door, or come to a blind bend, they will ask the DM what lies in wait for them. The party must then stop and decide what their next move is.

And here is a major difference between the fantasy war-gamer, or role playing gamer, and the Napoleonic and Middle Ages devotee. The latter attempts to recreate, in the smallest detail, a particular battle, or series of battles — and, if possible, to use their generalship to change the course of history. The RPG-er, as you will now be aware, creates his own world. Thus there are really no precedents, and each adventure is unique.

The DM, whilst taking no active part in the proceedings, nevertheless has profound influence on the game. Not only does he originate the dungeon complex, but he also populates the network with many monsters and traps for the unwary, as well as seeding the dungeons with plenty of treasures and other goodies.

Another major difference is the idea of puzzles inherent in the Role Playing Game — it is not enough to simply live a fantasy life by proxy. The player will, along with his colleagues, have to solve conundrums at every step, as well as fighting all the monsters along the way.

The combat system used when characters meet monsters and other nice people is complex, and much use is made of dice. Some, having more than the usual six sides, are called percentile dice, and give a degree of probability to the luck of the throw.

Along with all these differences, there is another, in that characters may become one of three main classes of characters. There is the class of fighting-men, which includes men, elves, dwarves and hobbits, magic-users and clerics. Add to these the spells that are a major part of the system, and you can see that this whole branch of war-gaming has departed quite a long way from our original tabletop game with lead soldiers.

We've spent a long time discussing RPG's, and indeed several complete books have been written on the subject. The whole game system has one major flaw — at least from the point of view of a crofter in the highlands of

Scotland, or the busy family man. And that is the time required to play, and the necessity of getting several people together at one time, prepared to devote a weekend to the pastime.

After the success of *Dungeons & Dragons*, it was only natural that many imitators sprang up, some making more impact on the scene than others.

Probably the most enduring of these has been Ken Andre's game *Tunnels & Trolls*. It simplified the rules of D&D to a great extent, but featured much of what had become so popular with the earlier game. Again, a Dungeon Master sets up a complex of caves (or in fact whatever sort of scenario he desires), and then the party of fellow-players is let loose to fare as they will. Combat is, similarly, moderated by the throw of dice, but *T&T* has no percentile dice. Spells are also inherent in *T&T's* system, progressing in hard-earned levels from the lowly 'knock-knock' (which opens locked doors) and 'take that you fiend' (which uses IQ as a weapon), through such quaintly-named spells as 'Zombie Zonk' and 'Mutatum Mutandorum' to the ultimate, 'Born Again', which pretty well speaks for itself.

However, the main development of *T&T*, which has endeared it to many thousands of fantasy game-players throughout the world, and which makes it particularly interesting to us computer users, is the unique system of solo dungeons which has taken *T&T* as its game system.

These dungeons take the form of illustrated books, each containing a ready-made adventure which can be played, according to the *T&T* rules, by one person. Really a series of multiple-choice actions, in which the text acts as Dungeon Master the books are a boon to the player who is unable to get together with other adventurers.

The great success enjoyed by these slim volumes is indicative of the great number of people who are unable to play full-blown RPG's, and who now look toward the computer as mediator, referee and Dungeon Master.

In the mid-1970s two enterprising chaps called Willie Crowther and Don Woods, whilst hunched over their huge mainframe computer, devised a game that they called *Adventures*. They were very likely *D&D* fans, as the scenario for their game included a complex of caves, peopled with assorted strange beings, and liberally scattered with treasure of all kinds.

Like *D&D*, the player makes his way slowly through unknown territory, receiving information about his surroundings — but this time from the computer. A tireless referee and Dungeon Master, the computer is the ideal medium for the fantasy game.

Other games came along in the wake of *Adventures*, probably the most successful being *Zork*, which is the forerunner of many of the adventure games implemented for the home micros of the late 1970s. The PET, Apple and Tandy machines were well-served with these for several years, while the original was passed, in disk form, around the circle of computer professionals. This free market was, however, rather black — the com-

panies which owned the big computers used by their employees for playing these illicit games, were, naturally, rather upset at this use of expensive computer time!

The obvious attraction of these computer-moderated adventures to the home enthusiast is that they can be played at any time, and alone, if need be (it's quite often illuminating to play these games with companions, each putting in their own contribution). Whilst the game can often take weeks or months to complete, the state of play can be saved to disk or tape and resumed at a moment.

The obvious drawback for the home enthusiast, is that in its original form, *Adventures* can only be run on a mainframe computer, costing a couple of hundred thousand pounds — not the usual living-room furniture.

With the advent of the cheap microcomputer, programs of the adventure genre came within reach of the home user. Now solo play became possible to the enthusiast not lucky enough to possess an IBM mainframe.

CHAPTER 2
The First Software

At the time of writing, mid-1983, a new microcomputer is announced every other week. Some of these have turned out to be what has been called 'vapourware', that is, it never materialises. Whilst this is a distressingly common phenomenon in the computer business, there are still machines about, and very good machines. The initial software package that eventually arrives from the manufacturer of the computer invariably includes an Adventure program.

Back in the pre-historic era of micro computing, about five or six years ago, three machines dominated the scene. All were American, and all were very expensive — and, incidentally, still are. All had implementations of *Adventures* and *Zork* written for them. The Apple I, the Commodore Pet, and the TRS-80 are still with us, in one form or another, but the cost of these machines in those far-off days served to keep the adventure club rather exclusive.

And then along came Sir Clive Sinclair. After single-handedly transforming the digital watch market with the Black Watch, and the pocket calculator market with the Executive (though only by virtue of their low cost and attractive styling, not long-term reliability), it seems, in hindsight, only inevitable that he should go on to concentrate on the computer market.

The ZX80 was, however, still aimed at the hobbyist, and commercial software was practically non-existent. It was not until the launch of the ZX81, with its attendant sales pitch at the larger consumer market, that home micro computer sales took off, and with that the creation of worthwhile software. Incidentally, this seems to be the exception that proves the rule that software helps sell hardware!

Adventure programs for the ZX81 are abundant and take several forms — some are purely text, while others are graphic in nature, with every shade in between. This follows the general trend in this area of software, with some authors staying with the traditional approach, and others treading a more innovative path.

We've seen the beginnings of computer adventuring in the Crowther/Woods original and *Zork*, but several other popular programs were written in the States, in the mid-1970s. As programmers became more efficient on their new toys, the ZX machines, so these older programs were adapted.

Many of the programs taken for conversion were originally published in David Ahl's *Creative Computing*, an American magazine. The most popular games included *Hammurabi*, often mis-spelt nowadays as Hamurabi. In the original, you, the player, have to direct the eponymous administrator of Sumeria in managing the city. Given so many acres of fertile land, so mny bushels of grain in storage, and so many people in the city, you have to balance all the ponderables in order to last a certain number of years. This type of game has been much adapted since its original appearance in *Creative Computing*, but the general term for the genre is management games.

Many of the adventure programs around today are actually descended from these management games, in their careful balancing of several ponderables.

Star Trek, first written in the late 1960s in the flush of enthusiasm for the TV series, is a kind of Hammurabi, involving, as it does, a delicate juggling between weapons control and ammunition levels, with damage controls and repairs all taking their toll. This sort of game is ideally suited to playing on a computer, leaving the machine to take care of all the details of galaxy scanning and status reporting. Leaving you to get on with zapping Klingons! The day may come, though, when someone writes a Star Trek program in which the aim is to make friends with the aliens...

Wumpus was also written many years ago, and has survived, in fact flourished to this day. The original game involved a search for the mythical Wumpus through a complex of squares. Using clues given to you by the computer, you eventually narrow down the choices, triangulating onto the final location of the beast.

We'll meet descendants of all these programs, *Adventures, Zork, Hammurabi* and *Wumpus*, in future pages. The early Sinclair machines, the ZX80 and 81, had many versions of the games written for them, and made commercially available on tape, or as listings in one of the many books written for the machines.

In particular, Artic Computing has produced an extremely worthwhile series of adventures imaginatively called Adventures A, B and C (well, they did have the alternative titles of *Planet of Death, Inca Treasure* and *Ship of Doom*). They have survived since the early days of ZXmania, and are very good adventures in the Crowther/Woods vein.

There are two reasons for their deserved success. First, the programs are, without exception, fiendishly difficult — but ultimately rewarding. Second, they are written in machine code (and this at a time when most software authors were still struggling with Sinclair BASIC), their responses to players'input being correspondingly rapid.

Representative of the graphic approach to adventure is a personal favourite of mine (and I make no apology for mentioning it!), J K Greye's *Catacombs*. The game draws heavily on both *Dungeons & Dragons* and *Wumpus* in its game mechanics.

As the player explores an unmapped underground complex, various monsters are met. If the threshold of a room containing a monster is crossed, information on the beast is given by the computer. Then battle is joined, and blows exchanged until the player or the monster is finally defeated. As higher levels are reached, the battles become harder.

Still available, *Catacombs* is a very good introduction for the ZX81-owner, to this kind of adventuring.

Another, rather novel, approach to graphic adventuring is seen in Foilkade's *Fantastic Voyage*. Based on the science-fiction novel of the same name, the action takes place in, of all places, the human body.

The book (and film of the same name), concerns the exploits of a team of neuro-surgeons who have been reduced to sub-miniature size in order to operate on their patient — from within.

The game on the ZX81 takes the form of a voyage around a map of the body, navigating your way through the maze of arteries and major veins. A map is supplied which may or may not help you, as the complex of blood vessels is difficult indeed to negotiate. Occasionally white corpuscles have to be shot at, arcade-style.

This is not strictly, I suppose, an adventure, but still it possesses the same kind of game mechanics as many more traditional programs in its search for the ultimate — in this case, the blood clot in the brain that has to be destroyed.

These are only two of many adventure games for the early Sinclair machines, but they all have the same disadvantages. Being written for the ZX80 and 81, they are all, necessarily in black and white — and mute. This is, of course, rather like saying that any black and white film is not worth watching, or any mono record is not good music. No, the classic programs will remain so, even in silent monochrome, and the lack of sound and colour may be, albeit rarely, a virtue.

The arrival of the Spectrum has bought a veritable flood of software, 95% of it games-orientated. A sizeable portion of this is adventure in one form or another.

Artic Computing have stuck with the same adventures that were written for the ZX81, merely transcribing them, complete with black text on white paper. At the time of writing, a new adventure has just been released — following tradition in being called Adventure E (and subtitled *Golden Apples*). Artic has, with this program, taken advantage of the Spectrum's features and presents the text in multi-hued ink on black paper — very colourful.

The game, like A,B,C and D before it, is purely text, but the new Adventure seems, at first playing, rather more traditional than the previous games. The program is written for the larger memory, so more detailed exploration of locations may be undertaken. It's a pity that 48K versions of the previous adventures could not be released by Artic — there would certainly be a large market for such an undertaking.

Other software houses have followed Artic in staying with the traditional approach, notable among these being Abersoft, Foilkade and Level 9. Whilst both Artic and Foilkade (with *Adventure 2000*, a reference to the number of locations in the game) have written their own adventure scenarios, both Abersoft and Level 9 have taken Willie Crowther and Don Woods' classic and translated it to the Spectrum. Both have, however, added touches of their own.

Level 9, for example, in their *Colossal Adventure* extend the original to some 70 more locations. And two sequels, *Dungeon Adventure* and *Adventure Quest*, follow on from locations in *Colossal*. The three together make an interesting suite of games, extremely tough, but always logical and amusing.

There are, of course, other programs that rely on text, yet, like Artic and Foilkade, do not follow the original ideas. Their distant relatives are usually *Wumpus* and *Dungeons & Dragons* (and sometimes, as in Mikrogen's *Sorcerer's Castle*, Snakes and Ladders!). These programs include such as *The Orb*, from Quest Software, *Velnor's Lair*, from Neptune, and *Volcanic Dungeon* from Carnell.

But, while these last programs may contain limited graphics, such as floor plans or weapon lists, to enhance the text, they can still be thought of as text adventures.

The Spectrum, however, cries out for its colour and sound capabilities to be used, and many authors have indeed taken advantage of these features in producing their adventure programs.

These generally take one of three forms. The first is an extension of the original *Adventures* idea, with a text input required from the player, but relying heavily on graphics for the program's appeal. Then there is the maze/Wumpus type. Finally there is the purely arcade adventure. This type makes often makes use of *D&D*-style characters and combat, and also the element of puzzle-solving, but transforms all this into a real-time arcade environment of gut reaction and hand-eye coordination.

Let's examine some examples from the various categories over the next few chapters.

The difficulty in describing the text adventure is in attempting to convey the essential flavour of the game, without giving away the answer to a problem that may have been intriguing a reader for many weeks — and much of the fun in solving text adventures is in shaking a tough problem by the neck until it succumbs.

We'll have a look now at *Dungeon Adventure* by Level 9. Any reader who finds himself currently in this particular adventure should skip the rest of this chapter! Those of you who may be thinking of buying this program should not despair — nothing crucial will be given away!

Dungeon Adventure is part of an excellent series written by Level 9, of High Wycombe, which starts with *Colossal Adventure*, based closely on Crowther/Woods original *Colossal Cave* program, but adding some 70

locations to the end game. The final program of the trio (there may be more by the time this book goes to press) is *Adventure Quest*. All inter-relate with each other and form a large fantasy world which can be explored and enjoyed.

The opening scene of *Dungeon Adventure*, and the puzzles associated with it, is a very pure example of the classic adventure program, and will serve to lay some ground rules for adventuring in general.

The booklet which accompanies the cassette of *Dungeon Adventure*, is typical of the quality of Level 9's output. A page of scene-setting, with a brief snatch of story concerning the seige of Minas Tirith, the death of the Demon Lord, and the protaganist's decision to rush off and search for the Lord's treasure in The Black Tower, prepares you for the start of the game. The rest of the booklet outlines the various commands you may give to the computer, along with several hints for the adventurer. Finally, a unique touch in an envelope in which you may, if you get dreadfully stuck, beg Level 9 for a clue.

At the start of the adventure, you, the hero, awake, cold and weaponless on a mudbank beside a large packing case, open at one end. There is also a piece of driftwood lying nearby. You see a stone bridge across a river, reaching from the granite cliffs above to the flat lands of the far bank. Now, you may, of course, go charging off up on to the bridge, eager to get on with exploring the caves which you secretly know are up there. And here we get to our First Golden Rule of Adventuring:

Look around every location in as much detail as possible.

Adventures of the traditional kind, which we are presently engaged in, almost invariably start off with the player alone and defenceless outside some sort of building.

In this present instance, there are some 170 locations to be explored within the cavern complex that you will find over that bridge. Your chances of surviving to see more than about half a dozen of them are, however, depressingly remote! Very soon after starting off, you will find certain objects become necessary — objects which can only be obtained from the outside locations. That is, the locations which you will find on this side of the bridge.

You will find out all this after a couple of tries at the cavern complex — and now you can start a search for these vital objects.

We know that there is nothing useful — yet! — to be found by going north across the bridge, so let's try going south.

But first of all, we must not forget our first Rule of Adventuring, and look around as much as possible. Now, if you remember, we are on a mudbank, and we can see some driftwood and a large packing case. Let's pick up the wood. Different programs recognise different commands for this action, and we can try PICK, TAKE or GET, the usual ones. In fact,

Dungeon Adventure will accept TAKE, so now we are the proud possessors of a lump of wood.

There doesn't seem to be much to do with it, so let's concentrate on something else.

The packing case, if you remember, was lying with one end open, so it would be a good idea to go inside, wouldn't it? NO, it wouldn't! Not yet, anyway! Writers of adventure programs are a sadistic lot, and would like nothing better than for you to walk wide-eyed and innocent into a trap — maybe just like this one. So let's try another key word which you will often come across in traditional adventures such as this one: and this is EXAMINE N (where N is the object to be EXAMINED!).

So let's type in EXAMINE CASE and see what the computer has to tell us. The case is enchanted! Not only that but there is room for us to crawl in. So we know now that we will not be eaten or stepped on by some passing monster, and we can proceed unscathed, for the moment.

Let's leave the case here, though. There are secrets to be learnt inside, of course, not least being the fact that the case is where you must deposit your treasure in order to gain points and win the game. The riddles to be solved in order to gain this information are fun to work out, and I don't want to spoil your fun at such an early stage.

Let's assume, then, that we've thoroughly explored the case, and are back outside on the mudbank, with our lump of wood. We have decided, maybe after several painful attempts, that we are on a sticky wicket in trying to negotiate the dark caverns to the north of the bridge. So let's try going to the south from the bridge. I am playing the adventure as I am writing, so this is a voyage of discovery for all of us!

We are now on an east-west road south of the river across which runs the bridge. A gigantic orc's head is carved into the cliff north of the river, its tongue forming the bridge. A ruined tower stands on top of the cliff. Exits are EAST, WEST and SOUTH. This is the standard formula for text adventures — enter a location and a brief description is given, with a list of possible exits. Actually, the descriptions of the various locations are rather detailed in Level 9's adventures, which is thanks to the larger memory available.

Usually, another list is also given — that of the objects and/or monsters or other entities to be seen, and the things that you may be carrying.

There is nothing to be seen lying around at this location, however, so we have to make up our minds which direction to take now. We have no clues here, so let's toss a coin and take the east road. The next location is, again, the east-west road, but now we are further along and we can see to the south a flat grassy plain, stretching as far as the eye can see.

This is one of those phrases that you will come to treat with respect. It usually means that if you once set foot in the location to which it refers, you will be lost forevermore — or at least until you give up in sheer despair!

So we will give that grassy plain a miss for the moment. However, we can

also see, so the computer informs us, a line of stepping-stones, which leads to a small island in the water. A young girl with flowing locks sits on the island — how charming! So let's type GO NORTH. The computer informs us that we are now on the stepping-stones, and the girl is still sitting there on the island. We type GO NORTH again. Oh dear! The girl was a siren, and sings her siren song — we flounder into the treacherous waters of the river. The computer informs us that we have managed to get ourselves killed.

The program now dumps us back on the mudbank, and we have to retrace our steps back to the east/west road, just south of the bridge. This time, let's explore to the east. We type GO EAST, and the computer duly tells us that we are now further east along the road, north of a steep, treeless hill.

This treeless hill seems to be worth investigating — going south, we come to the side of the steep hill, which rises into the clouds. Rumblings emanate from above! We can go UP here, so let's do so. We are now in a circle of distorted monoliths, etched into grotesque figures by the acid rain. Lightning arcs overhead, revealing the stark horror of this haunted place, while thunder echoes from every crag. Sounds very cosy, doesn't it? If we instruct the computer to LOOK around, a host of fierce flames will spring forth: they are Rakshasa! Their leader floats forward and challenges us to a game — if you win the throw of the dice he'll make our spiritual strength greater in some way. Of course, we may lose...

We have to play, and after all, it sounds quite a bargain, doesn't it? Unfortunately, we need a way of loading the dice (these Rakshasa are pretty cunning), and we haven't found it yet, although I might say that it is to be found somewhere in this opening sequence.

As you may expect, we lose this particular game, and once again find ourselves, after being asked if we want to play again, back on the mudbank — we're learning each time, though, aren't we?

Let's get back to that stretch of the road just north of the steep hill where we recently met the Rakshasa. We won't go back there again until we find the method of winning the dice game. So we will continue west to see what awaits us there.

And now we see a vast field of poppies which stretches west as far as the eye can see — and where have we seen that phrase before? We must tread very carefully here! A somnolent perfume hangs heavy on the breeze. Nearby we can see a dry poppy seed lying on the ground. Let's pick it up, by typing TAKE SEED. Now, let's retrace our steps — I have a strange feeling about that field of poppies!

We can work our way back along the road, past the steep hill, and the bridge, back to those stepping-stones. As we have a poppy seed with us now, we might be able to use it to fight that siren. At the stepping-stones, we type GO NORTH, which gets us onto the stones. Last time we were here, we got unceremoniously dumped in the water when we tried to proceed further north past the siren, so let's be a little cautious this time!

We might as well try the poppy seeds now — we'll type THROW SEEDS and see what that does. (I'm condensing a lot of heartache here, and a lot of effort in finding the correct solution!) The seeds, in falling, burst with loud explosions. Did they frighten the siren off?

To accomplish this we have to give a command to the computer. Type LOOK, and the computer will obligingly inform you of what is at the location. Quite often, of course, the answer you get is 'I see nothing special', but the question is, like EXAMINE, well worth asking. LOOK should not be confused with REDESCRIBE (REDES or R for short), which will instruct the computer to do just that with the present location.

Back to our present problems. Having tried LOOK, we find that the siren is still there, sitting by the river! Now we'll introduce yet another useful word — LISTEN. Unpleasant things have a nasty habit of hiding behind trees or rocks, out of sight of the computer, but can't help making slimy, squidgy noises, which will give them away!

Ah! Now we're getting somewhere! The computer tells us that we have been temporarily deafened, and can't hear anything. Although she must still be singing away as a good siren should, we won't hear her, so it's probably safe now to creep past her... but she's got us again! The temporary effect of deafness must be very short-lived.

Here we are back again at the mudbank. Now we can waste no time in getting back to that pesky siren, not forgetting to collect the poppy seed!

Another word worth remembering, and which would help us in the future, is SAVE. Most good adventure programs will allow this word which means that you can save your present stage of play to tape. You should, if possible do this before taking any risky initiative. We should have been more prudent and done this ourselves before attempting to get past the siren — we would now simply be able to type RESTORE and, once our old position was re-established, make a fresh attempt to cross the stepping-stones. As it is, we have to do a lot of typing to return, poppy seed at the ready, to that siren.

But here we are again, and we've dropped the seed, which duly explodes. We'll SAVE our position in case we are discombobulated yet again, and this time let's get on with it and rush past her. After all, we know now what effect the explosions will have. And — it works! The siren flees in panic, seeing that her song has no effect on us. In fact, the program will, at this point, allow you up to four commands before the deafness wears off.

And this is another ploy that the better adventure program will use. Approach a problem in one way, and the result may be completely different from the outcome of another attempt.

We're now at the southern end of a small island — the far end is occupied by a vicious-looking willow tree with six long rubbery branches. A silver mirror lies nearby. We can TAKE the mirror, but if we try and get past the tree, we will be killed. Why six branches? There must be a clue there!

Let's leave *Dungeon Adventure* now. This particular game is extremely

rich. We've only covered a few locations and I haven't mentioned the resurrection procedure which you will need before playing the game for real, and several other important details which can be found in the opening scenario.

Actually, this whole sequence, which consists of some thirty rooms is only an introduction to the main adventure, which takes place in the cave complex that we saw at the start. The original *Adventure* program started with the player standing outside a small building in a forest. This building contained several articles required for the adventure, such as keys, food and water — any treasure had to be brought back to the building. The very large network of locations at the start of *Dungeon Adventure* serves the same purpose as that small building in the original — but now in a far more complex way, which in fact is an intriguing mini-adventure in its own right.

After our first, fumbling attempts at *Dungeon Adventure*, you should have gained an idea of what it feels like to play a text adventure (and feel motivated to play Level 9's adventures yourself).

We've learnt several things in this chapter which help you when playing a standard adventure. In the next chapter we take a look at arcade adventures.

CHAPTER 3
The Hobbit

The Sinclair Spectrum, with its colour and (very basic!) sound, understandably has had an awful amount of arcade, zap'em, software written for it. Without wishing to enter into a discussion of the state of play in the microcomputer world, it is nevertheless obvious that the vast majority of micro-users concentrate on reaction games — and why not?

After our short analysis, at the start of the last chapter, on the two styles, text and graphic, of adventure programs, let's turn now to the graphic.

And within this category itself, we find two distinct sub-categories.

The first is best represented, at the time of writing, by *The Hobbit*, by Melbourne House.

The Hobbit is probably the best known, at least by reputation if not personal experience, of the adventures available for the Spectrum. Already a classic, it is still the most talked-about and puzzled over. In the true tradition of software progress, other programs will arrive that will, no doubt, improve upon *The Hobbit*, but none can now usurp its place as the first truly great adventure for the Spectrum.

The Hobbit, by J R R Tolkien, is, if you like, the preface to the monumental *Lord of the Rings*, probably the richest source of material for the adventure writer. This vast work, with its three volumes of absolutely riveting story, and supporting historical documentation in *The Silmarillion* (together with encyclopaedic information from authors such as Tony Tyler, with his *Tolkien Companion*), is an absolute goldmine of ideas. Melbourne House pulled off a fabulous coup in acquiring the rights to use *The Hobbit* from the Tolkien estate, and has produced a program which lives up to the novel. *The Lord of the Rings* is an obvious sequel to *The Hobbit*, and is the one that everyone is, of course, waiting for. No news is presently available on the subject.

The basic concept had been used several times before *The Hobbit* was written — a text adventure in the good old style, supported by graphics. Melbourne House has taken many scenes from the book, and illustrated them with beautiful high-resolution pictures, using an impression of the pictures, drawn by Tolkien himself, supporting the text in the original book. This was a logical progression from the earlier programs from other writers which featured block graphics. The other important details that set *The Hobbit* apart from previous text/graphics adventures is what Mel-

bourne House calls Animation and Inglish. Animation means, in this context, that the characters in the program, such as Thorin, Gandalf, and the various monsters, like Gollum, the Butler (he didn't do it, by the way), and the wood-elves, carry on their own lives while you, as Bilbo Baggins, get on with the adventure. In practice,this means that characters keep walking in and out, singing about gold and so on. Gandalf has even been known to walk off with the front door to the hall at Bag-End! All this is not necessarily useful, but entertaining all the same.

Inglish (and that is no misprint!) is Melbourne House's term for compound sentences.

Now let's imagine that you have reached the Dragon's lair. He is, of course, guarding the gold which you have battled through many dangers to retrieve from his clutches. When you arrive, you want to draw your sword, fight the dragon, and then collect the gold. Most programs would require you to type in several commands before the final objective was achieved. So you might well have to type: "DRAW SWORD", "KILL DRAGON", then finally (and if you were successful!), "GET GOLD".

The Hobbit however, will accept commands such as: "DRAW SWORD AND KILL DRAGON, THEN GET GOLD AND LEAVE", thus saving finger stress, but also having a more natural feel.

The adventure may be solved in many ways. Unlike the traditional programs, in which there is only one solution to the complete puzzle, *The Hobbit* may be completed by any one of several methods. A percentage SCORE is given throughout the game, and people have finished the adventure with scores as low as 42%, and as high as 210%! The aim remains the same, however, to rescue the gold from Smaug, the dragon, and return to the comfort of your own home under the hill at Bag-End.

So whichever way you choose to solve the game's ultimate puzzle, there is a different game awaiting you should you wish to play it another way.

As an aside, let me say that I feel that *The Hobbit* is, as I said, a classic, but nevertheless, a flawed classic. Many people have, for example, reported that pressing CAPS SHIFT and 1 together (and this could well happen accidentally!) causes the program to crash. A similar fate awaits the person trying to enter certain locations without a particular item. Animation (Melbourne House's name, if you remember,for the character's self-motivation) occasionally manifests itself in strange ways — kill off Gandalf (surely you wouldn't be tempted to do that, would you?) and he may well turn up alive and well at some later date (and may also be carrying your front door). These are little quirks that Hobbit-ists have become accustomed to, and in fact there is almost a fan club of Hobbit bug spotters!

So *The Hobbit*, whilst being, on the surface, a traditional adventure, nevertheless plays in a unique way. The puzzles set by programs in the classic style (programs like Level 9's) have one solution (usually!) which remains the same. Once you have completed the adventure, that's it. Of

course, the puzzles may take months to solve, and several programs have vast areas which act as a kind of divertissement — all those locations off the beaten track of the game, which play no part in the solution, but which would serve to entertain you. *The Hobbit* itself has an intriguing location which is, the computer informs you, "TOO FULL TO ENTER". Does this means that the Spectrum's memory is too full for the programmers to write more locations? Almost certainly not, for they would surely be more inventive than this! Is there, more likely, a plan afoot at Melbourne House to launch as a sequel carrying on from here into the uncharted area, à la Level 9?

There has been, and there will, I hope, still be in the future, lengthy discussion on *The Hobbit*. If you have wondered whether to buy this most charming, frustrating, annoying, beautiful program, rest assured that you will enjoy many hours of good adventuring!

CHAPTER 4
Graphic Adventures

Although *The Hobbit* is the most accomplished, there are many adventures which support their text with graphics of some kind or another. There are, however, others which are pure graphic programs.

We can determine two rather obvious categories of graphic adventure. The first is descended from *Wumpus*, a game we have met before, whose popularity can be ascribed in large part to David Ahl of *Creative Computing*. We'll look in detail at a program of this category in a moment. The arcade adventure, the other main variety of graphic program often relies on *Dungeons & Dragons* for its combat mechanics, but can really be bestowed with any scenario.

To see what we might expect from a version of Wumpus, let's look at *Sorcerer's Castle*, written by Mikrogen. The scenario concerns Murtceps (that name look familiar, for some reason!), the sorcerer with the eponymous Castle. The program, whilst setting up the large castle, goes into some detail about your mission, which is to find Murtcep's Great Crown. While the set-up procedure is going on, the computer counts down with the aid of a large on-screen clock — this way you know that the program has not hung-up on you.

After this initial set-up, the player is asked to choose which character he wants to play from a list of options. He may be:

a dwarf
an elf
a fighter
a hobbit
or a wizard

Then the player is asked if they are male or female. I have to admit that the effect that this question has, on the running of the program at least, seems to be minimal.

But depending on what the answer is to the character list, points are given out to the player in three important departments. Thus, if the player chooses to be a hobbit, he is given 26 points, in the following way:

strength	6
intelligence	10
dexterity	10

Should he decide to choose an elf as his alter ego, the points are distributed slightly differently:

strength 6
intelligence 8
dexterity 12

The fighter, as you would expect, has a lot of strength, and quite a high dexterity. Following the good old D&D tradition, he has, unfairly, a very low intelligence:

strength 14
intelligence 4
dexterity 10

And so on. However the player decides, he is then given six extra points, to allot as he likes (and if you press a wrong key, the computer has an answer — for example the fighter will find the computer sneering at this IQ of 4).

Then on to the local armoury. This has a list of assorted bits of armour and weapons that you might like to take with you. You have 60 gold pieces, and may choose from (and the prices are in gold pieces):

armourplate	30	swords	30
chainmail	20	maces	20
leather	10	daggers	10
lamps	20	flares	1 each

Let's play *Sorcerer's Castle*. We have decided, after looking at the list of options, to become a fighter, and we have used our six extra points to end up with our attributes distributed thus:

strength 15
intelligence 6
dexterity 11

And we have purchased from the armoury:

chainmail
a lamp
a mace

Once equipped the adventurer can now enter the castle of Murtceps. The program tells you:

YOU ARE NOW ENTERING THE CASTLE
YOU ARE IN ROOM 1–4 ON LEVEL 1
YOU HAVE:- STRENGTH 15

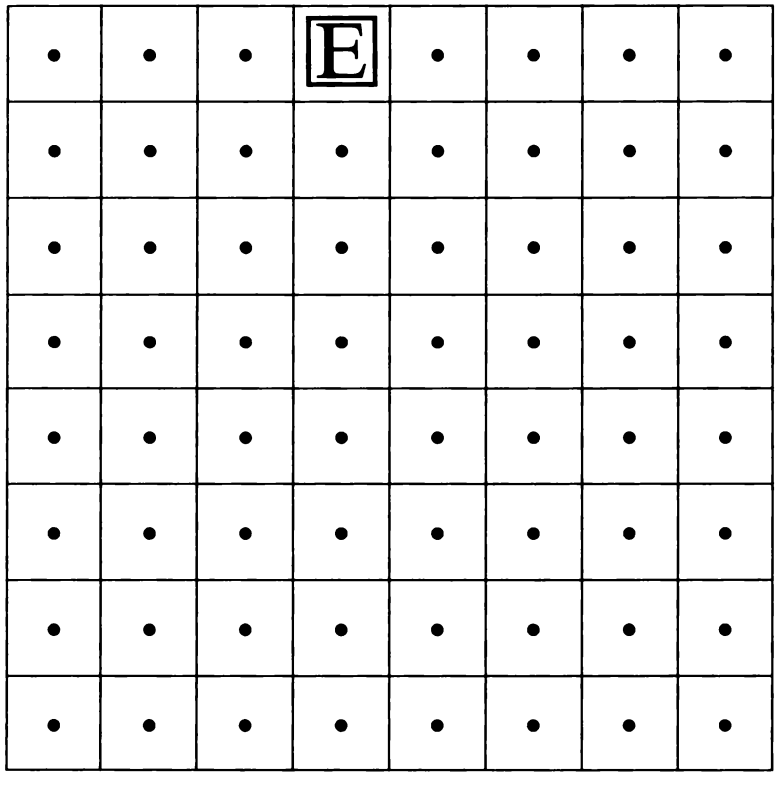
INTELLIGENCE 6 DEXTERITY 11
YOU ARE WEARING CHAINMAIL AND CARRY A LAMP, A MACE 0 FLARE + 0 GOLD PIECES
HERE YOU FIND THE ENTRANCE
WHAT DO YOU WANT TO DO?

Now is a good time to press H, which is the HELP key. Unlike the text adventures we have seen before, this does not result in a cryptic remark from the computer, but a long list of possible options for the player to take. No hours of head-scratching to find the right word to invoke a response from the program!

The HELP list shows up, amongst many other options, that we can type just the first letter of the direction that we want to go. There is also a (M)ap command, so let's enter M, and see what happens.

The computer answers with a map of the present level:

You are in room 1,4 on level 1



The map shows us that we are in a complex of rooms, or dungeons, or what you will. We are currently at top centre of the network, in the square marked E, which is the entrance to the castle.

What do we do now? Well, we could, of course, go charging off in any old direction willy-nilly, but... if we did that some monster would probably rip our silly heads off, so let's pause awhile and think the situation over. If you remember, at the start of the game we were given the option of buying several weapons. Amongst these was that innocuous item, the lamp. We may, imprudently, have passed over this particular weapon, having spent all our money on swords and maces. Hard luck if we did! The lamp, in this game, as in the others of its kind, is one of the most important weapons in our arsenal. Without the lamp we cannot see what is awaiting us in adjoining rooms, and must therefore go blundering along in ignorance of what awaits us.

Luckily, we had the foresight to purchase one of these extremely useful items. To use it, we must, according to the HELP menu, type in (L)amp. If we do that, the computer answers with:

"Where will you shine the lamp"

As we are starting off, it doesn't really matter where we shine the lamp — all locations are unknown to us. We remember from our sneaky look at the map that we are currently on the northernmost wall of this level of the castle so we can't move northwards anyway. Just to see the effect, however, of this, let's type N. The program responds:

"Your lamp shines out of the castle door"

We can't go that way (like it or not, we are now in the castle for good), so we have to go to the south, east, or west. Let's toss a coin and shine our lamp to the (W)est. The status report appears again, as it does at every location, and we are also given the information that we can see gold pieces in the next room. We might as well have these, as we may well need some money later in the game. As we are going quite well in this direction, we'll keep on going west, not forgetting of course, to shine our trusty lamp ahead of us.

The computer tells us that in the next room to the west is a Warp, whatever that might be. Until we have found out what that is, we'll leave it well alone. Let's see what's to the south.

The computer tells us that in the room to the south is a chest. That doesn't sound too bad, so let's type S(outh) and see what it's all about. Now we are in the room with the chest — what shall we do now? The program is waiting for our answer, so we type in H(elp) to see the menu of options. Halfway down is

O to open a chest

Let's do that. The computer replies with:

YOU OPEN THE CHEST;
IT EXPLODES!

It goes on to give your status, as it always does, but, unfortunately, says no more about the effect of the exploding chest. Maybe it has weakened us in some way,but no more is said about it.

We'll carry on, undismayed, and see what might await us to the south. Shining our trusty lamp into the room, we receive this message from the computer:

YOUR LAMP SHINES INTO ROOM 3–3
THERE YOU SEE THE SILMARIL

The Silmaril is a jewel created by Feanor, an elf of Middle Earth. Actually, there were three of them, so maybe the other two are to be found elsewhere, within Murtcep's Castle. The jewels are very powerful, so it must be worth getting them. We'll go south to get this one.

And the computer tells us that:

YOU NOW HAVE THE SILMARIL

So it is obvious that we merely have to move into a room to get the treasure that it may contain. Let's keep moving south, we've had pretty good luck so far, apart from the chest. And now we see more treasure! Something called the Norn Stone, whatever that may be. Let's get it — all this treasure feels good, doesn't it?

We'll try a different direction now, and shine our lamp westward. Waiting for us in Room 2–4, so the computer informs us, is a Bear. If we move into this room we are given a warning by the computer that there is danger about, by means of a very tuneful warble. We are told what monster is presently here,and given a little menu of options:

(A)TTACK
(R)ETREAT
or
(B)RIBE

We merely have to type in the first letter of the option we choose. Let's have some fun and type in B.

Oh dear, (or words to that effect)! The only way that the Bear will allow us to escape, is if we give him the Silmaril. And we've only had it five minutes! Ah, well. Let him have what he wants, we may well find the other two Silmaril soon. Having escaped his clutches, we can carry on with our trusty lamp held high before us.

To the east, we can see a Crystal Orb. That sounds as if it should be interesting. Moving in Room 1–4, we only find the Crystal Orb. Taking a peek at the Help menu, we can see that the command to use is:

(G)aze into the Orb

The Orb tells us that it sees us being killed by a Wolf! Obviously we have to stay out of the way of Wolves for a while.

From here, we can see, to the south, stairs going down. To the east, we see — yup, a Wolf. So south has to do for the moment. Actually, when you enter a room with stairs, you have a choice of going down, or up, or just passing through. We'll pass through this time, as we are having quite good fun on this level, without braving the unknown deeper in the castle.

We look again to the south, and here we see, in Room 1,6 a sinkhole. I'll use my inside knowledge of *Sorcerer's Castle* to tell you that entering a room with a sinkhole in it will deposit you at some random room at some level deeper in the castle. This would appear to be a good time to stop!

Sorcerer's Castle is an excellent text-based version of the old *Wumpus*. The castle complex contains many weird and wonderful things that I haven't mentioned. Things like the occasional Book, which may impart great knowledge, or may turn out to be a booby-trap, which will stick to your weapon-arm, thus reducing your effectiveness in a fight. Or the flares, which will momentarily show you the contents of half-a-dozen rooms around you. Or the Vendors, who will try and sell you, at vastly inflated prices, new armour and weapons (or may even attack you!).

To see how *Wumpus* may be interpreted with the addition of graphics, let's turn now to *The Valley,* from Argus Specialist Press.

Originally published as an extremely well-documented listing in the magazine *Computing Today, The Valley* is now available for many of the popular micros. The program's author is Henry Budgett, editor of *Computing Today* — or at least, must owe a lot to HB for its development. Although written in mid-1980 the program only shows its age in one detail. It does not, unfortunately, AUTO-RUN on Loading, which is extremely unusual nowadays. Although full instructions are included, within the packaging, on how to get started, this is an oversight that should be remedied.

When RUN, the title page informs you that this program is:

THE VALLEY (48K)

If you didn't have the expanded Spectrum, you would not be reading the title page!

After this, however, the program gets much better. A couple of pages of instructions are given, giving the player the movement keys (the usual cursors, and combined keys to give diagonal movement). After this, a menu of

characters is presented. The mechanics of the game depend heavily on those from *Dungeons & Dragons,* as well as from *Wumpus*, and this can be seen from the character list, from which you may choose to be:

a wizard
a thinker
a barbarian
a warrior
or
a cleric

The character you choose at the start will determine the game's tactics. Wizards, for instance, will have much more Psi-strength than Barbarians, who will, conversely, have a greater combat strength. So the original characters must be chosen carefully. As in any conventional role-playing game, players will have their favourites, and there is a nice little character-saving routine which will allow you to continue the game over several sessions with the same character — or try different characters to see how each type fares.

Incidentally, if you feel a bit mischievous and press a key other than those specified in the menu of characters, you get labelled as a DOLT, and dolts don't last long in *The Valley*.

The scenario concerns the land of Tybollea, saved from the Selric hordes by Princess Evanna, and the mighty wizard Vounim, who together forged strong magicks to vanquish the foe. In gratitude, Evanna bequeathed to Vounim The Valley, which lay between the two Castles. Vounim, however, fell into the ways of evil, and Evanna eventually had to find some way of ridding her land of the wizard, who had now built a temple to the Lizard Goddess Y'Nagioth.

Evanna could find no-one to help her in this task, and set off on her own to vanquish the evil wizard.

Alarian, a young wizard of Evanna's court, had been apprenticed to Vounim, and gave the princess an amulet, studded with six precious stones. The amulet had magic powers, and could bestow upon the wearer life after death.

Bearding Vounim in his lair, Evanna vanquished him. The evil wizard's might was considerable, however, and the brave princess succumbed to his great magic. Though not before she had hidden the amulet in the Temple of Y'Nagioth and distributed its six stones about the six floors of the Black Tower. Her final act was to transfer her magic to her Helm, which she then hid in Vounim's Lair, where she perished.

Peace reigned over The Valley at last, with Alarian the young wizard acting in the dead princess' stead.

Now, ten thousand years later, the evil wizard Vounim is attempting to regain his hold over The Valley. Alarian, as you might expect, is rather

physically weak by now, but has bought together a small band of adventurers to help him find the amulet with its stones, and through them, the Helm of Evanna. Only with the Helm can he conquer Vounim once and for all.

That is the sort of scenario which you will find time and again as preface to many adventures — it's rather nice, don't you think? The original article contained some two and a half closely-printed pages of this stuff. Do all computer magazine editors have virulent imaginations?

The program now sets up the main screen graphics. A "safe path" is created at random, though it will generally wind from the left hand side of the screen from the "safe castle" there, across the valley to the right hand side.

Dotted about the playing area are forests and swamps, and The Black Tower. In the forest you will find Vounim's Lair, while the swamp contains the dreaded Temple of Y'Nagioth.

Below the map is your status report, showing the amount of treasure you have collected, your experience, number of turns, combat strength, psi-power and stamina.

You may move along the safe path without harm. However, one of the main aims of the game is to gain experience points, by fighting monsters, and thus progress through 28 levels, from lowly "MONSTER FOOD" (although you actually start at the higher grade of "PEASANT"), all the way to "MASTER OF DESTINY", encompassing along the way "HERO", "SUPERHERO", and "PRINCE OF THE VALLEY" — not bad for a mere computer freak, is it?

So, you could stay put on the path, but you would get absolutely nowhere! In *The Valley,* utter caution is rewarded no more than reckless foolhardiness.

Move off the path, however, and the adventurer is fair game for all kinds of wandering monsters.

And the monsters are really vile! Here you will meet such lovelies as Balrogs, Orcs, Water Imps, Ring Wraiths and Fire Giants amongst many others, with one or two really HEAVY monsters which you would do well to avoid — and about which I am certainly not going to tell you!

If, and when, you stumble across a monster, you may be lucky enough to catch it by surprise, and get in the first blow. You do, though, have the choice of attacking or retreating. There is no bribing in this program — the monsters only want one thing, your blood! This game is in real-time (this means that everything happens at the same speed as real life), so when the computer urges you:

"STRIKE QUICKLY!"

it is no idle command. If you don't respond immediately, you will have lost the initiative, and the monster will get in the first blow — and this holds true throughout the combat sequence.

Three choices of physical attack are open to you — you may strike at the body or limbs of your enemy. And you'll find that this method of attack is accurate, but takes longer to wear the beast down, allowing him more time to slash away at you! Or you may elect to go for the head, which is not so accurate, but usually fatal when it connects. If you miss the head, of course, your adversary is likely to duck under your arm and get in a sneaky blow.

If you get a bit fed up with slogging it out with your sword, you may call up a spell. There are three available, although not all at once. The lowest-ranking spell is SLEEPIT, which of course stuns the monster, thus facilitating your escape. This spell may be used by anybody, regardless of psi-power, or experience points. PSI-LANCE requires a minimum of 49 psi-power and 1000 experience points — moreover, it may only be used against psychic monsters. The best spell, tantalisingly called CRISPIT, can be used only if you have more than 77 psi-power and a whopping 5000 experience points. A word of warning — make sure that a spell is available to you before you attempt to use it.

In moving around the open countryside of *The Valley*, you will meet many monsters (we've had just a glimpse of them), but you'll also find places of great magic, and ancient power, which boost your psi-power, or combat strength. Conversely, you may stumble into a circle of evil, which will deplete both your stamina and your magic. Of course, you may also come across a hoard of gold, which will add to your treasure, and make you a little richer.

All this wandering about is a good way of collecting experience points. If you feel strong enough and fast enough, you can wander around for as long as you like. The program will keep seeding the valley with new monsters for you.

In the end, though, you'll want to get down to some serious treasure-seeking, and this can only be achieved in one of the mysterious buildings.

To enter, for example, the Temple of Y'Nagioth (which is, if you remember, where the dying Evanna secreted Alarian's amulet), the player must head for the Swamp. As his character-token moves onto the square occupied by the swamp, the screen clears, to be replaced by the swamp. The temple is situated somewhere near the middle of the swamp, surrounded by a moat.

The adventurer has first of all to navigate the boggy marshland, with (of course!), its attendant creatures, and then the moat. Here, a different kind of fiend is met — Water Imps and deadly Krakens and so on. Having survived these, we hope, the player may then enter the Temple, whereupon the scene changes once again to show the floor-plan of the Temple. Again the player moves around at will, meeting monsters, some of them unique to this location. The other buildings too, have their own peculiar monsters.

In the buildings, however, the location of the treasures may be seen by

the player, and he will naturally head toward them — after all, in the Temple, one of those treasures will be the Amulet, which is the first objective of the game. Some of them, though, will turn out to be mere worthless baubles. They are all, anyway, situated a long way from the player's starting point!

In *The Valley,* as in all good adventures, text or graphic, the various parts of the game fit together like well-oiled machinery.

The player will, first of all, have to find the amulet, then go on to the Black Tower of Zaexon to find the six stones to fit the Amulet. These can only be found if you have the Amulet, and must, moreover, be found in the correct order. The documentation that comes with the tape, exhaustive though it is, does not tell you what that order is, although the original article did (hard luck if you haven't got a copy!). Only after doing all this (and don't forget, you'll be fighting every metre of the way), can the intrepid traveller then go on to Vounim's Lair, in the forest, to find the Helm of Evanna.

The Helm, however, will only be found when you have achieved the rating of War Lord (which is 25 on our scale of 0 to 28). As it will take many hours of play just to get out of the Cannon Fodder classification you can see that the Save Character and Game routine is essential.

The Valley is an almost perfect fusion of many of the elements that go to make up this type of Dungeon and Dragon-orientated game.

We have, first of all, a good combat system, with adrenalin-pumping action, good graphics, good monsters and, a detail that escapes many authors, a built-in high score table in the ratings — a character can be built up over many hours of play, from lowly Monster Food to Master of Destiny. Players will find themselves rooting for favourite characters, who become stronger and more potent as each level is reached.

The interlocked tasks set the player in *The Valley* are a necessity for the good adventure program.

Only one small point could be improved upon — the monsters themselves are not shown graphically, although full information on each one is given below the map of the playing-area. It would be nice to have a little picture of the Ring-Wraith as it belts you.

One main advantage of this program, for us budding adventurers is that it is written almost entirely in BASIC (except for a small sideways scrolling routine at the start). REM statements are included in the program, and they are extremely clear — even to showing the graphics that will be built up from the BIN statements.

Having survived Vounim's Lair, and the Temple of Y'Nagioth, not to mention the six horror-filled floors of the Black Tower, you probably think you can sit back and rest. Not a bit of it!

Now try Crystal's *Halls of the Things*. Not six, not seven, but eight (count 'em!) floors of maze-like corridors crammed with utterly blood-

thirsty creatures (or Things).

The packaging is excellent — a stout, rigid plastic box with an excellent slip-in cover with full-colour artwork houses the cassette. The instructions for play are on the reverse of the sleeve.

While the program is loading a screen is presented to the player, containing all the protaganists of the game — you (a little humanoid), the Things (various shapes, pyramidical, triangular and so on), Treasure (piles of golden nuggets), and Elixir (bottles of stuff).

The game is pure arcade, and in my opinion, one of the best for the machine.

The scenario is, unlike *The Valley*, rather sparse, but again, we have an excellent balance to the game. The top seven floors contain, somewhere amongst them, seven rings. These must all be found before entry to the final, eighth floor can be gained. On this floor is the key, which will give you entrance to the drawbridge, which is the way across the deadly acid-filled moat which is your only exit from the tower.

The scenario is, however, unimportant, as this program is pure arcade. Following the author's recommendation, the beginner should start at the highest level, which is the easiest. Climbing the stairs you will see the energy barriers to all the floors (except the lowest). Whilst on the staircase, your little man is perfectly safe, but stepping through the barrier will unleash the denizens of hell.

Each level is a maze-like complex of corridors, patrolled by bloodthirsty creatures (the eponymous Things). In this program, there are no choices, as in *The Valley*. You are not asked whether you want to (a)ttack, (r)etreat or (b)ribe. These lovely creatures want one thing only — your blood! And they have an uncanny knack of lurking around corners waiting for you to walk into their trap. You can, if you like, wait for them to finally make the first move, but beware — more Things will creep up on you as you are waiting.

Now, if you attempted to hunt for the Rings without weapons, you would not survive long (about three seconds, actually), so the authors kindly supply you with several.

The main weapons available to you are magical. Fireballs are used singly, and have the amazing ability to seek out the monsters even round corners, so they're pretty potent — but they use up a lot of your magic, and may only be used one at a time. A lot less draining on your powers, yet maybe just as effective, are the Lightning Bolts. These are best used in multiple salvoes, and they rebound throughout the neighbouring corridors, scouring the area for Things. You'll see their remains lying, smouldering, where they have been incinerated.

The other weapons are non-magical, and should be used to preserve your magical strength. First of all, you have many arrows. To use these to full advantage, you should cautiously poke your nose around a corner. If there

is a Thing nearby, it will then loose off a Lightning Bolt or Fireball at you (yes, they have the same deadly magic as you, I'm afraid). Having thus determined the presence of the enemy, the player then sets the cursor key in the required direction, and swiftly moves out into the corridor, loosing off a salvo of a dozen or so arrows. Effective at short range, the arrows should make short work of any nearby Thing. If you find magic projectiles still being hurled your way after your flight of arrows, you can loose off more arrows, or hurl a few Lightning Bolts, which should do the job.

If you get really desperate, you may use your sword, but if you are close enough to use this, you will be sustaining some pretty awful wounds.

By pressing the 1 key, the action is halted, and you are allowed a look at the Status Report. This will show you the amount of magic you currently can call upon, the amount of Things you have killed, the number of arrows you have left, and the amount of wounds you have suffered. This is important, because, in *The Hall of the Things*, there is no such cissy thing as five lives, not even three! No you only have one life, and this will be terminated when your wounds reach 100% on the Status Report. By pressing H you may heal some of your wounds, but this uses up Magic, thus cutting down on the number of potent weapons left to you.

There is also a report on the number of rings that you have found so far. A high score is present in the program, and devotees of the novels of Philip Jose Farmer will recognise the name of the high scorer, which is Kickaha. For all I know, there may well be a fanfare of trumpets and a choir of angels when this high score is beaten — it is 100, and I have so far managed 35! Apparently, the game has only been completed once, and that is by the game's authors.

But all this is as nothing, when compared to the sheer addictiveness of the game. *The Hall of the Things* has that essential simplicity, once all the keys are learnt, that will keep the player coming back again and again to play it.

So we have now seen four programs that illustrate the main types of adventure available for the new machine. In the next chapter we will meet some of the inhabitants of these games.

CHAPTER 5
Meet the Cast

We can start having some fun now, and look at the monsters, treasures and locations that might be met in the typical adventures.

Monsters

The term monsters, for our present purpose, can be applied to any character in the program that is out to do us harm — and they are not always recognisable as such!

Like the programs themselves, we can see distinct categories here. The classic adventure, descended from the original mainframe *Adventures*, contains fairly passive monsters which tend to sit there, waiting for some brilliant strategem from the player to scare him away, as the snake did in our introductory scene at the start of the book. Or we may have to avoid the monster by finding a way around him.

The evil dwarves are a notable exception in the *Colossal Cavern* adventures. They appear occasionally to throw a weapon at the adventurer. The first dwarf throws an axe. This must be picked up by the player and then thrown at the succeeding dwarves. They are all hurling knives, after the initial axe, but if you remember to keep retrieving the axe, you should have no trouble in surviving their attacks.

In Level 9's version of the classic, an endgame is included that extends the original to some 70 new locations. And here you can really get your own back on those little dwarves! By dropping dynamite near a crowd of them, you can score many points. Incidentally, during this finishing sequence, you can also score points by saving from death a number of elves.

As played originally, on mainframe computers, after hours, the printer was often the only means of seeing what was happening — so blow by blow combat in *D&D* style was not really feasible.

This leads us to the next monster category, which we find in the action, or arcade games. In this category, the monsters are extremely active, and definitely out to get you! The combat system from the role playing games like *Dungeons & Dragons* is often used in these games, as exemplified in *The Valley*. And the monsters themselves, as befits the ancestry of this type of game, are of the leg-ripping, skull-crushing, heart-stopping type commonly met in RPGs. There follows a list of some of these lovely things,

along with brief details of their attributes, and origins. I've put them in a very subjective order of fiercesomeness — so, if you meet a Balrog, you would, usually, treat it with rather more respect than, say, an Orc.

Monsters!

Dwarves

Appear in the original mainframe classic, hurling knives and axes at the player — nuisance value only.

Orcs

The Jack-of-all-trades monster — appears in most Tolkienesque adventures. Extremely vicious and ugly creatures, in groups, and usually wielding spears or scimitars. There is an Orcish Archer in one of Level 9's games.

Waug

This appears in *The Hobbit*. Originally an extremely dangerous creature, who often accompanied Orcs on their raiding trips, the Waug doesn't seem to be active in *The Hobbit*. Its mission in that game appears to be lying around dead.

Snake

Most programs featuring a snake do so to create a problem rather than to actively cause a player harm — see the first chapter.

Birds

Often to be found at the top of mountains sitting on Golden Eggs. Find a way to frighten them off before attempting to purloin the treasure!

Dragon

Very frightening and powerful — do not approach unless very strong, or very sure of how to cope with it. In *The Hobbit*, you may need help.

Elementals

Come in four types: Air, Fire, Earth and Water. You will need magic of a sort relating to the Elemental before attempting to fight.

Vampires

No need to tell you how to fight these! Before coming across them, you should have picked up any requisites at other locations (don't turn your nose up at that garlic!)

Mind Vampires

A special sort — they are not after your blood!

Dogs

Very lowly monsters, but often attack in packs. Only the most rudimentary of weapons needed to fight them.

Wolves

Even more vicious than Dogs.

Ghosts

You won't need a sword or spear to battle these!

Werewolf

Only magical weapons will be of any use against these.

Goblin

Small, ugly creatures. Delight in prodding their victims in the knees with sharp sticks.

Hobgoblin

Larger, more dangerous and more cunning than their more animal-like halfbrothers.

Harpy

Winged creature of amazing strength and agility.

Siren

A sea-faring harpy — usually found basking on rocks near the sea, singing. Rock music was never like this.

Troll

Devious, greedy, oafish creatures. You'll meet two very typical examples in *The Hobbit*. They don't like the sun!

Barrow Wight

Ghostly apparitions which populate the Middle Earth wilderness.

Centaur

Half horse-half man, often armed with bow-and-arrow, of which they are masters.

Fire Imp

Little, agile flames of nuisance value.

Fire Giant

Large dangerous flame.

Thunder Lizard

One of the most potent of monsters. Only tackle it if you are well-armed with conventional weapons, and have a fair amount of dexterity.

Sand Worm/Purple Worm

Extremely dangerous! The sand worm is basically a mouth with a 60-foot stomach behind it. The purple worm is similar but with eyes, and not confined to the sand.

Minotaur

The well-known bull-like creature. As dangerous as you would expect a highly-intelligent bull on the rampage to be.

Wyvern

Another winged beast, this one comes equipped with vicious fangs and claws.

Balrog

One of the most dangerous of all monsters, and often met in adventures, as typifying the whole Tolkienesque, *D&D* world of fantasy.

Water Imp

More nuisance.

Kraken

Water dragon.

Lich

You'll have occasion to meet one of these later, so be warned — they are extremely powerful. They are former wizards who have died and brought themselves back to life to wreak havoc.

This is, of course, only a partial list of the monsters you may expect to meet. Most adventure programs will feature some of these, and more of their own. In writing your own adventure, your imagination can be the

only restriction. Just about anything can be pressed into service in a game, and I have seen everything from snowmen to London double-decker buses being used as death-dealing enemies.

Most programs that rely on a *Dungeon & Dragon*-style combat system will keep you informed of your current physical status. This may take the form of physical points, combat points, or food points. You may, indeed, be given a combination of all these, but it will be clear to you when playing, that a decision on whether or not to fight with a particular monster must be made by you, taking into account your own strength, using whatever system the program adheres to, together with your own knowledge of the monster's own rating.

Of course, you may be given no forewarning of a monster's presence, and thus have to fight whether you want to or not.

This sort of program would be a very unfair one, and they're not common. Most games will give you both a certain pre-knowledge of the monster's capabilities and a method of escape should you wish to decline the challenge. You may not, though, make good your escape, if the monster is a particularly speedy one!

Spells

The sort of combat we've spoken about so far, with the system of strength or combat points is, of course, based on the physical side of combat — that is, a few well-aimed chops at the monster's head with your broadsword or morningstar. Many programs, however, give you the options of employing magical powers, and casting simple spells at a monster. This idea is very well-entrenched in *D&D, Tunnel and Trolls*, and other RPGs, where many pages of the rulebooks are devoted to complex spells, which become ever more complex the longer you survive as a wizard.

The basic idea of a spell remains the same, however — to ZAP the monster with a well-aimed spell. This can take the form of a simple "SLEEP" spell which lays out the monster for a certain amount of time, to the "CRISPIT" spell as seen in *The Valley*, which annihilates the thing completely. Spells of this potency require a great degree of experience to use.

Unfortunately, the programs currently available for the machine do not, as a rule contain very good, or complex spells. In fact, *The Valley* is one of the most spell-conscious programs around, and that, as we've seen, can only boast three spells.

Another program which features spells, is *Velnor's Lair*, from Neptune.

The program concerns the black wizard, Velnor, who has gone into hiding in the Goblin labyrinth of Mount Elk. Your task, as the adventurer, is to prevent Velnor from metamorphosing into a demon on earth, which event will not be good for mankind. You must explore the laby-

rinth, facing his ghastly guards, and finally come face to face with Velnor himself. You start the quest armed with a torch, a tinderbox, and a club.

At the start, you are asked whether you want to be:

A Warrior which will, of course, be your choice if you fancy a bit of monster-bashing. He has no spell-casting powers, but may use any magical items found in the labyrinths.

A Wizard who is a bit of an easy target, as he has very limited physical prowess. His strength lies in his spells, of which he has three:

a) *Polymorph-* this costs him one spell point (he starts with ten in total), and changes any non-magical, living creature into a harmless frog. The success rate is normally 50%.

b) *Teleport* which costs three points, and which will set you, with whatever you may be carrying, back at the entrance to the labyrinth.

c) *Fireball-* the most potent, but also the most costly, at five points. This spell kills any creatures, without magical protection, at the location.

A Priest who is somewhere between the wizard and the warrior in ability. Like the wizard, he has three spells to choose from, and an initial 10 spell points. The spells available to him are:

a) *Shield* which costs him three points, and which makes him less vulnerable to attack — it can, however, only be used once.

b) *Dispel undead* which does just that, and also costs three points.

c) *Heal* which costs four points and heals all the priest's injuries.

After all this preamble, the rest of the game follows our *Wumpus* model fairly closely. A maze of tunnels is explored by the player. Many problems have to be solved, and, of course, there are lots of monsters to battle with. Apart from the spell routines, which are complex and worth obtaining the program for, *Velnor's Lair* is a run-of-the-mill adventure, but one which should give some pleasure to the *D&D* enthusiast.

The fact remains, however, that the player looking for a computer adventure for this machine, which includes very complex spells, as in *Dungeons & Dragons* and the rest of the role playing games, will, frankly be disappointed. It's entirely possible, though, that a magic adventure will be released the week before this book!

Weapons

At the start of the classic adventure, you are weaponless. You have to find your own armoury during the course of the game, and it will probably con-

sist of not much more than a short sword, or dagger, or axe — and these are usually thrown at you by the occasional passing dwarf.

These weapons come in handy when dealing with the dwarves, and you'll get points for killing them, but you will probably not get much of a chance to use them against dragons, or sirens, or other monsters. No, they are to be beaten by guile — and you'll find that the Big Bad Pirate who, every so often, rushes in to steal all your hard-won treasure and rushes out again to hide it, long before you can throw any axe you may also be carrying!

So we have to look elsewhere if we are to use a lot of lovely, blood-dripping swords!

Although the Dungeon and Dragons system, with its swords, maces and so on, is a fertile ground, well-used by game-writers, there are several adventure programs available for the machine which make use of more conventional weapons.

Richard Shepherd has written several adventures for the Sinclair machine — some of them use the term rather loosely, but two of them are fairly good examples of adventures. The text adventure is represented by *Superspy*, while the graphic is represented by *Transylvanian Tower*.

The plot of *Superspy* is not entirely unadjacent to that of a James Bond movie, and combines text and graphic adventure. The first part is pure text, with several puzzles and anagrams to be solved. Once you have solved these, the graphic, arcade second part of the game may be tackled. The weapons available to you at the start of the game are what we are interested in at the moment, though.

These are very different from the usual sort of weapons that are usually met in adventure programs. In *Superspy*, you are offered, at the start of the game, a small list of modern weapons, consisting of:

pocket bomb
cyanide gun
concealed knife
Walther PPK pistol
digital watch

My personal favourite is the Walther PPK — very reminiscent of the James Bond stories. You are allowed to choose three weapons from this list, and with these must battle the various enemies that you'll meet during the program. The opening section of the game is a rather simplistic adventure, relying on code-breaking and anagram-solving for its solution. Once the first part is behind you, a maze game is begun, in which you must enter Dr Death's island and find his lair. In the maze you will come across Jaws, and, if you are lucky, escape him!

Transylvanian Tower, another of Shepherd's adventures, is described by him as 'a chilling 3D adventure, with spectacular graphics' — a little reasonable hyperbole, as the graphics are, as in *Superspy*, written in BASIC. The Maze, which is the core and whole point of the game, is of the 3D variety, as you may gather from the blurb. 3D is one of those terms that has become accepted computerese after long use, and in this case means that you, seemingly, stand in a maze, looking down a corridor, or, as in the present program, at three walls of a room, together with any doorways that might be at the location. The drawing of the room, in *Transylvanian Tower* is done in BASIC, so is a little slower than machine-code would accomplish. The difference is roughly similar to the difference between a blink and a heartbeat, though, so is quite acceptable.

The object of this adventure is to beard the Count in his Tower. As you prowl about the various levels, trying to find your way around, you'll find various treasures lying about that may (or may not!) help you in the final confrontation.

The weapons available to the player are:

a knife
a dagger
a sword

These are the obvious ones, but there are also magical weapons, which include:

a silver cross
a mirror
a magic ring
a magic cloak
a magic apple
and, of course
a clove of garlic

without which, a Dracula adventure would not be complete!

On the first level of the five to be negotiated, there are no weapons to be found, and none needed. On the succeeding floors, you will find all the objects — and also bats! A certain number of these must be killed on each level in order to progress to the next. In order to achieve this, the player is granted a bat-zapping pistol, loaded with ten bullets. The bat flutters about while you are trying to draw a bead on it, and if you are successful in shooting it down, lies in a crumpled heap on the floor of the room. As a last gift before dying, each bat will give you a floor plan of the present level, with its multitude of rooms to be negotiated. However, if you run out of ammunition before entering the next level (and you may find other bullets lying about the castle), you are transported to a room, at random, somewhere on

the same level. And you might find this is a sealed room! There is a way to get out of this, but a certain weapon, or treasure is required.

Once the Count is met, only one of the weapons will conquer him. You are not told which. Well, you wouldn't want things too easy, would you?

So the weapons you might meet in adventure games are varied in the extreme — weapons can be made of anything, and are really only limited by the writer's imagination.

Fighting monsters with deadly weapons, however, is not really the ultimate aim of the traditional adventure game — sorry Jason, Fred and Tom of 2C! The monsters are usually protecting something, and that is usually

Treasure (sometimes heavily disguised).

We've finally arrived at what is without doubt the real reason for putting up with all the frustrations of playing this wonderful game.

Earlier in this book, we looked at *Colossal Adventure*, from Level 9. This adventure, based, in the first part, on the Crowther/Woods mainframe original, contains many treasures to be collected during the course of the game. Each of these is worth a certain number of points, and in the text, or text/graphic game, the score is the thing.

Take a look at this situation. You are in a long corridor, in a complex of caves. You have arrived here after collecting treasures and other objects. To find out what you are presently carrying, type INVENTORY (or usually, just INV.). The computer replies:

YOU ARE CARRYING:

> The Magic Helm
> The Golden Ring
> The Brass Shield
> The Comic
> The Fire Opal
> The Brown Gloves
> YOU ARE IN A LONG CORRIDOR. TO THE WEST YOU SEE A DARK CAVERN.
> WHAT NOW?

Well, we might as well go in — this can often prove dangerous, as the author will probably be testing us, and will set all sorts of traps about his complex of caves. The next message is often:

> YOU HAVE SLIPPED IN THE DARK AND BROKEN YOUR SKULL. DO YOU WANT ANOTHER GAME Y/N?

But we have been warned in advance of the lack of light in the cave, and we have not been told of any light source in the corridor, so it is a reasonable assumption that there will be light of some sort in the cave itself. Having first SAVEd our position (any reasonable program will allow you to do this before any risky steps are taken, thus making it easy, if killed, to quickly resume), we decide to take the risk of looking into the cave. We type:

W(est)

and the computer answers:

YOU ARE NOW IN THE DARK CAVE. A FAINT LIGHT GLOWS FROM THE ENTRANCE BEHIND YOU. BETTER GET SOME LIGHT QUICK.
WHAT NOW?

Try this:

READ COMIC (for some light relief, geddit?)

Ingenious, but no good, I'm afraid! The computer merely repeats its message:

ITS DARK IN HERE. BETTER GET SOME LIGHT.

This is not too good! Adventure programs often let you stumble about in the dark for only a short time, or a couple of turns before deciding that you haven't the faintest idea of how to proceed and dumping you unceremoniously in the nearest ice-cold underground stream, or causing you to bang into an overhanging rock. Whatever method it chooses, the program will often draw the game to a sudden conclusion.

But what's this — something else comes up on the screen:

YOU FEEL SLIMY TENTACLES WRAP AROUND YOUR LEGS.
IT IS A SMALL OCTOPUS.

Now the octopus may come in handy here. So:

GET OCTOPUS

And the computer come back with:

UGH — NO THANKS — ITS ALL SLIMY
WHAT NOW?

WEAR GLOVES

O.K. I AM NOW WEARING THE BROWN GLOVES
WHAT DO I DO etc.

GET OCTOPUS

I HAVE THE OCTOPUS. MANY HANDS MAKE LIGHT WORK. A SOFT LIGHT EMANATES FROM AN INVISIBLE SOURCE. YOU ARE IN A SMALL CAVE. THERE IS A SMALL WOODEN BOX HERE. TO THE NORTH YOU SEE AN UNDERGROUND STREAM FLOWING THROUGH A SMALL CRACK IN THE ROCK. THROUGH THE CRACK YOU CAN JUST MAKE OUT SOMETHING GLITTERING IN THE DARK BEYOND.
WHAT NOW?

He!He! Some more treasure! Must get it! Flushed with justifiable pride at our success at using the octopus, we attack the next problem with gusto. How are we going to get at that gold in the next cave?

We can try the direct route, by typing:

N (for NORTH)

YOU CAN'T GO THAT WAY
WHAT NOW?

That's a bit of a blow. The computer, darn it, doesn't seem very impressed with our previous brilliant deductions, and is already putting more problems our way! There must be a way through the crack. Maybe we are carrying too many goodies, and we can't squeeze through.

DROP ALL

O.K.
WHAT NOW?

N

YOU CAN'T GO THAT WAY
WHAT NOW?

How about jumping in the stream, we might be able to swim or float through.

JUMP IN STREAM

YOU MUST BE JOKING — I CAN'T BREATHE
UNDERWATER!
WHAT NOW?

Hmm, this is getting silly. Is there anything to help us in the cave? There's a little pile of treasure on the floor which we dropped a few moments ago, but none of that is useful to us now. Wait, there's a box, isn't there. How could we have forgotten that?

EXAMINE BOX

YOU SEE A SMALL WOODEN BOX.
THERE IS NOTHING UNUSUAL.

Computers can be very pedantic!

OPEN BOX

O.K. THE BOX IS OPEN. INSIDE YOU SEE AN AQUALUNG.

Aqualung? Underground stream?

GET AQUALUNG

YOU ARE CARRYING:
THE AQUALUNG
THE BROWN GLOVES WHICH YOU ARE WEARING
THERE IS ALSO A SMALL BOX WHICH IS OPEN

That's a lot of stuff to carry! Wait a moment, though, the box might be useful. We're told it is small but maybe it will carry everything we want to take with us, and maybe it will float!

Now we can try:

GET RING

O.K. YOU HAVE THE RING
WHAT NOW?

PUT RING IN BOX

O.K.
WHAT NOW?

LOOK

and we get the same list as before, but with a difference at the end:

AND A SMALL BOX WHICH IS OPEN. IT CONTAINS THE GOLDEN RING

It works! Now we'll have to repeat the process with the rest of the treasure. After several moments typing, another "LOOK" informs us that the box now holds all our hard-won treasure. The octopus, we keep for the moment. Now let's try:

THROW BOX

WHERE?

IN STREAM

O.K. THE BOX IS CARRIED SWIFTLY FROM VIEW INTO THE CRACK IN THE ROCKS

Now let's do the same with the octopus. The computer tells us that the octopus has also disappeared from view. Now we'll type:

JUMP IN STREAM

and the computer tells us:

YOU MUST BE JOKING — I CAN'T BREATHE UNDERWATER!

So boring! We'll have to go through every motion for the computer's benefit.

WEAR AQUALUNG

O.K. YOU ARE WEARING THE AQUALUNG.
I AM SUFFOCATING. QUICK DO SOMETHING!

Whoops! Forgotten something!

TURN ON AIR VALVE

O.K. THE AIR VALVE IS NOW ON.
WHAT NOW?

JUMP IN STREAM

O.K. YOU ARE CARRIED ON THE SWIFT CURRENT OF THE STREAM THROUGH THE CRACK IN THE ROCK, AND INTO A CAVE. THE OCTOPUS HAS GONE BEFORE. MANY HANDS HAVE MADE THE LIGHT WORK. ABOVE YOU IS THE BANK OF THE STREAM.

We breathe a sigh of relief. Now we must get up onto the bank of the stream.

U (for UP)

O.K.
YOU ARE IN A LARGE CAVE.
YOU SEE A SMALL BOX. IT IS OPEN AND CONTAINS MUCH TREASURE.

(but wait a moment before going to retrieve the treasure!)

YOU ALSO SEE:
A DRAGON ASLEEP. HIS SCALES GLISTEN IN THE LIGHT.
ALL THAT GLISTENS IS NOT GOLD! HE IS STIRRING.
WHAT NOW?

What a question! Well, what do we do now?

This is an example of a typical problem to be met in a traditional adventure (I won't tell you which one, in case you are currently playing it) — a series of interlocking problems that eventually yield a result. That result may, as in our example, lead you into a deal of trouble, but of course the possibility is that you will end up with treasure.

Getting the treasure is not, however, the final story, as you have to return to a certain location to deposit the treasure and collect your points.

Whilst this may seem a bit of a chore, as you will have to renegotiate all the perils on your return, there are, in most good adventures, certain routines to bypass this necessity.

Crowther and Woods wrote a certain magic word on a cave wall in their adventure — utter this at the right time (that is, when you have an armful of treasure you want to get rid of), and you'll find yourself, treasure included, back at your original starting point. You may then obtain your score. Other adventure programs actually allow you to carry the treasure-repository along with you! You may have to drop it at some point to allow you to collect treasure (or squeeze through narrow cracks in rocks!) — so don't forget where you left it!

Let's have a look at some typical treasures. And, as with my list of monsters, there will be a (rather loose) progression of potency.

Treasure List

Helm	The boring iron sort of helmet
Silver helm	A bit better
Golden helm	Quite nice
Magic helm	This is more like it, although there is an even better one later

gemstones	opals, diamonds of course, emeralds, topaz, lapus lazuli — you name them and you'll find them somewhere in an adventure.
Ring	Again, the boring sort.
Silver ring	Not bad.
Golden ring	A bit common nowadays — every adventure seems to have one! The one found in *The Hobbit* is famous for having no perceivable purpose!
Invisible ring	Doesn't look much (how could it?) but if it makes the wearer invisible too, then it is obviously useful.
Invisible cloak	This is even better than the ring.

These last two lead us to a secondary list — of 'found' weapons. That is, weapons that the explorer will find in his travels, scattered about the caverns, rather than the sword that most adventures provide their player with at the start of the game.

Sword	The standard weapon of almost all adventures, although some allow you to stumble upon Swords in your wanderings.
Torch	A very basic weapon, useful for clearing an area of ants, frightening wolves and so on.
Club	A bit of a brutal weapon — not very subtle.
Mace	Another basic weapon.
Dagger	A secretive variation of Sword.
Kris	Just one example of exotica. The typical adventure author is extremely imaginative when it comes to dreaming up new weapons, and you can meet all sorts of devices you never knew existed. Who said this game wasn't educational?
Scimitar	Another bit of exotica on the face of it, but actually one of the favourite weapons of the Orcish tribe, so often met in adventures.
Staff	A piece of wood!
Silver staff	A piece of silver, which has magical connotations.
Stake	You can probably work out what this is for (if not, just wait for a vampire).

Silver sword Like the Silver Staff, this is a bit more magical than its more mundane cousin!

Silver bullet Useful if you have a gun!

Fire whip This is just one example of a weapon that can be used against an Elemental. These are creatures formed from the very elements (Earth, Fire, Air and Water) — and as such, they can only be fought with appropriate weapons.

Meanwhile, back at the treasure:

Book Generally just worth points, as an artefact, but they are understood to be repositories of ancient lore, so, on opening, may increase your Intelligence. But beware, they have been known to explode on opening!

Gold coins The standard currency in all the best adventures, but not as valuable as

Gold which is usually found in 'Hoards', and is often the objective of the adventure.

Crystal Orb Representative of many magical items to be found in adventuring, the Orb allows the gazer to see into the future, or another location, thus seeing his death.

Helm of immortality This is v-e-r-y potent, and should not need explaining — actually, if you find this, you have probably won the game (not to mention the universe!)

And finally, but not leastly, the ultimate treasure of any adventure to date,

The Golden Sundial of PI To date, mid-1983, this £6,000 work of art has not been won, but then there is only one time and place in which it is to be found, and that time may well be 1986. It is still, however, the only treasure featured in an adventure program which can actually be won by the player.

That concludes our very short list of the treasures you may expect to meet in the typical adventure — by no means exhaustive, it nevertheless should give a rough idea of the sort of object that will crop up time and time again.

The better authors, of course, employ a good deal of imagination in creating treasure, weapons and all the other paraphernalia in their adventures.

PART 2

The Eye of the Star Warrior

CHAPTER 6
Why a Graphic Adventure?

Now at last we're getting down to some serious programming!

The following section of the book will describe a graphic adventure, and its development. Many of the modules described can be taken and used in your own programs, and the techniques discussed, adapted as you wish.

Before starting, let's consider the question — why did we choose a graphic program? After all, the original classic adventure is text-based, as we've seen.

There are two reasons — the more important, probably, is that in this way we can present the maximum number of techniques. As I've said, you may wish to adapt these to your own programs (or, indeed, add your own touches to the present program). The second reason is simple — a text adventure, while being a lot of fun to play, is not much of a surprise after being typed in from a listing!

Having said that, though, you may wish to add to your own text adventure, and the techniques used in *The Eye of the Star Warrior* may be used just as easily in this situation. For instance, the section on generating the room complex will be just as valid in a text game, as will the movement routines.

First of all, however, let's have a bit of history, which might help explain what you're doing in this hellish dungeon!

CHAPTER 7
The Legend

There is a dimly remembered legend of the third continent that tells of a treasure with such power that it threatened to consume all life on earth. The Eye of the Star Warrior contained at its heart the fire of a thousand suns!

The legend dates back to early in man's civilization, before the Great Flood. The Demon, Agor, escaped from the negative zone and took a mortal form on Earth. Disguised as the Wizard Domire, Agor held the lands of the third continent in tyrannical rule for over two million years. No power on Earth could match Agor, and all who opposed him, perished. The wisest of the wizards of the third continent met in secret council, and the Brotherhood of the The Star Warrior was formed. They plotted to use mystic arts forbidden by the Creator. The Brotherhood sought to breathe life into a clay giant, and to give this artificial man power greater than the Demon Agor. Only in the depth of space did such power exist! Combining their psychic strength, the Brotherhood set a thousand suns on a collision course that would last two million years.

Future generations protected the giant's body and nurtured the growing gem that would eventually sit in the Star Warrior's single eye socket. As the time grew close, many members of the Great Council questioned the plan to give the Star Warrior so much power. Aldous, last leader of the Council, gave orders for the eye socket of the Star Warrior to be lined with explosives.

When the meeting of the suns took place, the energy of their collision was channelled into the magic stone. The stone was then ceremoniously placed in the eye socket of the Star Warrior, and it gave life to the clay giant.

The Star Warrior was terrible to behold! It strode across the countryside, burning the land and destroying everything in its path. Soon, the Star Warrior met Agor in battle, and the Demon was incinerated. The Star Warrior placed himself on Agor's throne, and a new reign of terror spread across the third continent. Under orders from the Star Warrior, members of the Brotherhood was hunted down and slain. Aldous was captured and brought to the Temple, there to be dismembered by Trolls, under the terrible gaze of the Star Warrior.

As Aldous' right hand was severed from his arm, it rose from the floor,

and fired a lightning bolt that struck the Star Warrior in the eye. The explosives were detonated and the eye fell from its socket.

Without the power of the stone, the giant was no more than a statue, and it crashed to the ground. One of the Trolls uttered a curse and brought down his sword towards Aldous' chest. Before he died, Aldous threw a magic field of energy around the stone, to prevent it being replaced in the Star Warrior.

The stone has since fallen into the hand of evil Wizards. It is now possessed by a Lich, a dead Wiz ırd with almost demonic power. He is close to breaking the shield around the stone.

The Eye of the Star Warrior must be destroyed before the powers of evil can use it once more!

CHAPTER 8
Create your own Dungeon!

Our first task, then, is to create the dungeon complex.

The complex will be spread over three levels, with a maximum of 300 cells, 100 on each level. It is quite possible to have one large level only, consisting of the same 300 cells. It would get rather difficult to find your way about after a while, though, and in having different levels, we can more easily set different tasks, and varying degrees of difficulty.

Now, although we are eventually going to be moving about a system of rooms, and will then be calling them rooms, at this point in the program's development, we will be talking also of cells.

In a dungeon complex, we will naturally require a number of individual rooms, through which we can move, and which will contain monsters, and (we hope), treasure. These rooms will be inter-connected (unless you want your player to tunnel between rooms).

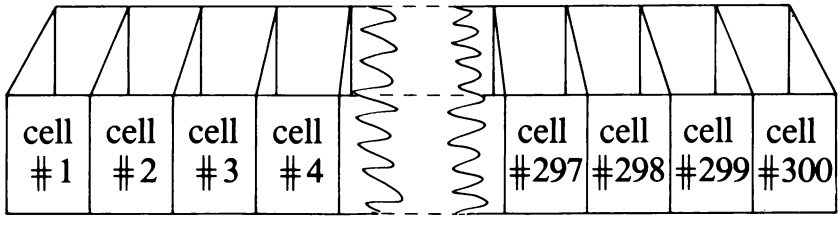
Dungeon set-up

```
1634 REM **********************
1635 REM SET UP DUNGEON
1636 REM **********************
1640 DIM C(300)
```

LEAVE THE COMPUTER SWITCHED ON, OR SAVE THIS LINE!

This line simply DIMensions the array C to 300. C now contains 300 locations, and can be thought of as a long length of boxes, if you like, stuck together. There are 300 of them (see **Figure 1**). This takes care of our maximum number of cells.

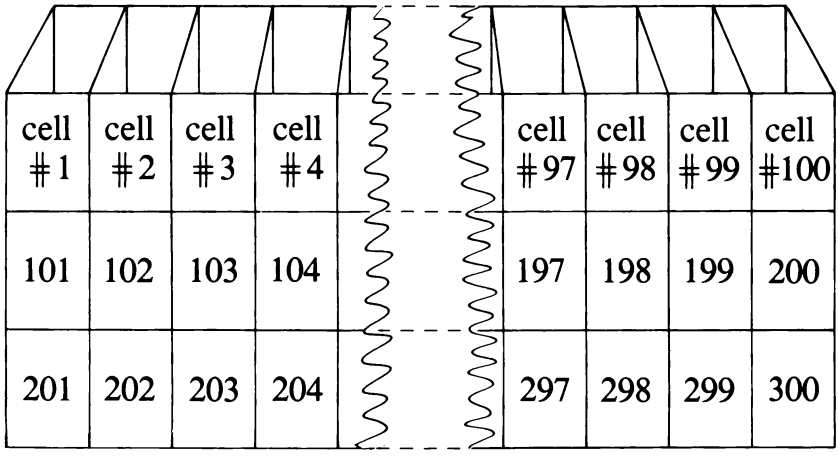
Figure 1



Now, we could leave the set-up there — we would be able to move from cell #1, to cell #2, and then onto #3 and #4 and so on, or indeed back to cell #1 if we wished — but what a boring game. In a text adventure, we would be moving in one direction all the time — no variety! And if a terrible monster was barring our way, we would be able to progress no further, being able to find no way round him. So, we have to provide a bit of flexibility.

To do this, let's first imagine (and this will be entirely in our imagination, as the computer is still thinking of the array C as one long line of boxes) that we can somehow bend our 300 boxes, or cells, into three layers of 100 cells each. If we chose to stay with this format, we would at least be allowed some freedom of movement up or down from any given cell, and also left or right from a cell to an adjacent cell.

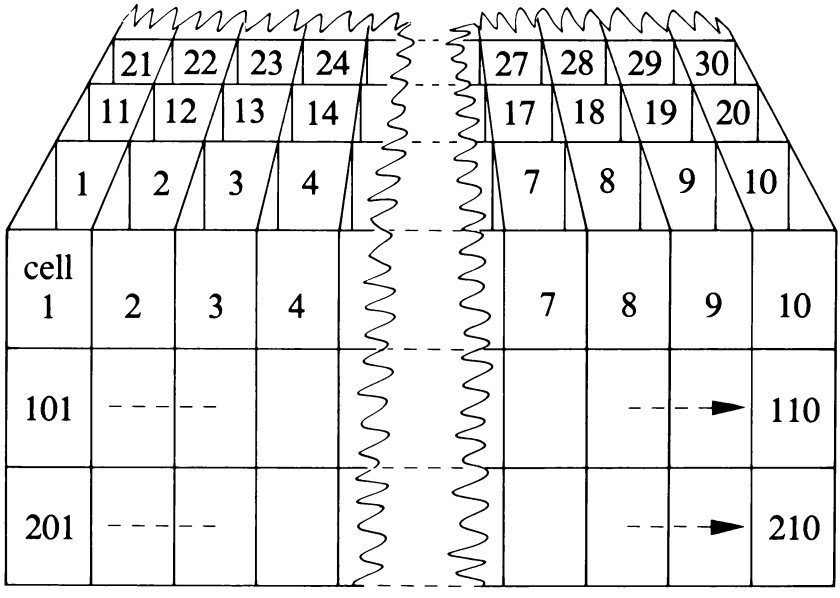
Figure 2



But, although this is a bit better than our first set-up, you will have noticed that we still only have two-dimensional movement. We could make our matrix, which at the moment is 3×100, into a matrix of, say, 30×10, or some such construction, which would make for a more complex set-up, but still only two-dimensional. Let's go three-dimensional. Staying with our three levels, we'll construct a box, with the dimensions $10 \times 10 \times 3$. Our complex now looks like **Figure 3.**

So now we have three levels, each of 100 cells (10×10), in three dimensions, about which we may roam to our heart's content. This is the theory — but how does the computer hold this information? In fact, computers are not able to grasp concepts like three dimensions, as are we humans. Array C is still, to the computer, a long line of 300 boxes, or locations, and will remain so, however we may choose to imagine it.

Figure 3



How, then, does our computer know when we are on level 1, or 2, or 3?

Let's look against at **Figure 3**. You can see from this that the player, in real life, could simply go from cell #1, directly down to cell #101, and thus be on the second level. But the computer is still, as we said, holding the whole dungeon complex in one long array of 300 boxes. In fact, **Figures 2** and **3** exist only in our imagination. To see the reality of the computer's view, turn again to **Figure 1**. It would need some pretty fancy coding to achieve easy movement between our three levels. The way we get our computer to distinguish between the three levels, is by putting an impassable barrier, or wall, between a cell of one level, and a cell of another level. In other words, design and create a wall.

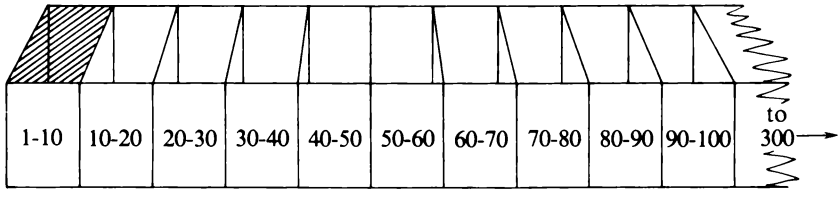
Now we can return to the computer! We have 300 cells, or locations, in array C, and we know that we can move freely around all these cells (if we had keyed in a movement module — we will later!). We have to now make some kind of barrier between the levels, and the best way we can do this is to place a number in the required location in array C, which will act as a marker. As we have 300 locations, we need to have a number higher than this as a marker. We could make it 301, or 426, or 872 — it doesn't really matter, but for the sake of clarity in the listing (not to mention that in six months time, we may have forgotten what one of these numbers stands for!

Let's type in:

```
1650 FOR a=1 TO 10: LET C(a)=999
: NEXT a
```

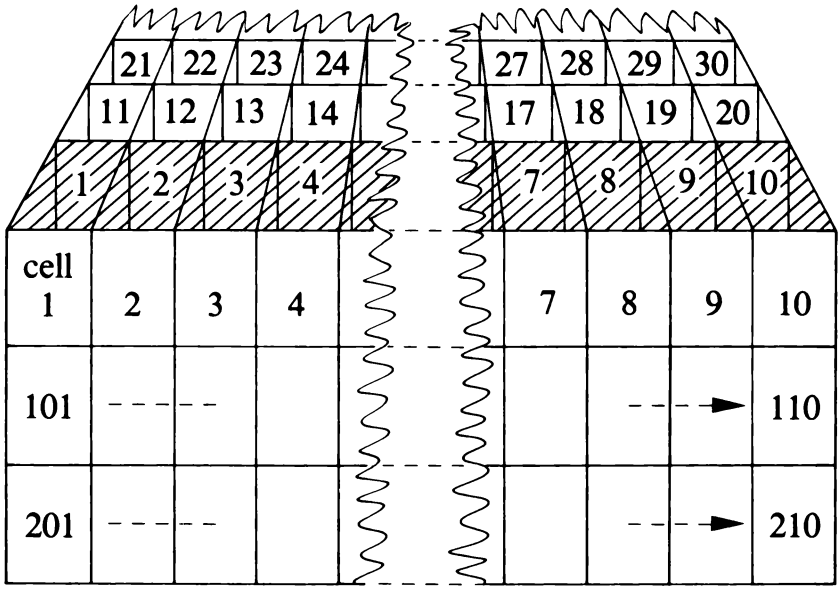
This line will create a block in cells 1 to 10, ie the array C will now contain, in its first 10 locations, the value 999, which effectively bars the player from visiting these particular cells. Let's see what this looks like to the computer.

Figure 4



Why cells 1 to 10? Look now at our imaginary three-storey dungeon.

Figure 5



Aha! An impenetrable wall to the outside. Cells 1 to 10 are now no-go areas, and will never contain monsters or treasure. The next three lines do the same for the other levels.

```
1660 FOR a=90 TO 110: LET C(a)=9
99: NEXT a
```

Line 1660 creates an impassable barrier at the last row of Level 1, and the first of the second level, while

```
1670 FOR a=190 TO 210: LET C(a)=
999: NEXT a
```

creates the wall between the second and the third levels. Finally

```
1680 FOR a=290 TO 300: LET C(a)=
999: NEXT a
```

creates the last barrier at the end of the third level. Now, we need to put in the boundaries at the "edges" of the levels

```
1690 FOR a=11 TO 291 STEP 10: LE
T C(a)=999: NEXT a
1700 FOR a=10 TO 300 STEP 10: LE
T C(a)=999: NEXT a
```

These two lines create blocks in steps of ten, thus creating the remaining boundaries.

Look at **Figure 6**, a plan view of Level 1 to see what has happened. The other two levels have been treated in the same way.

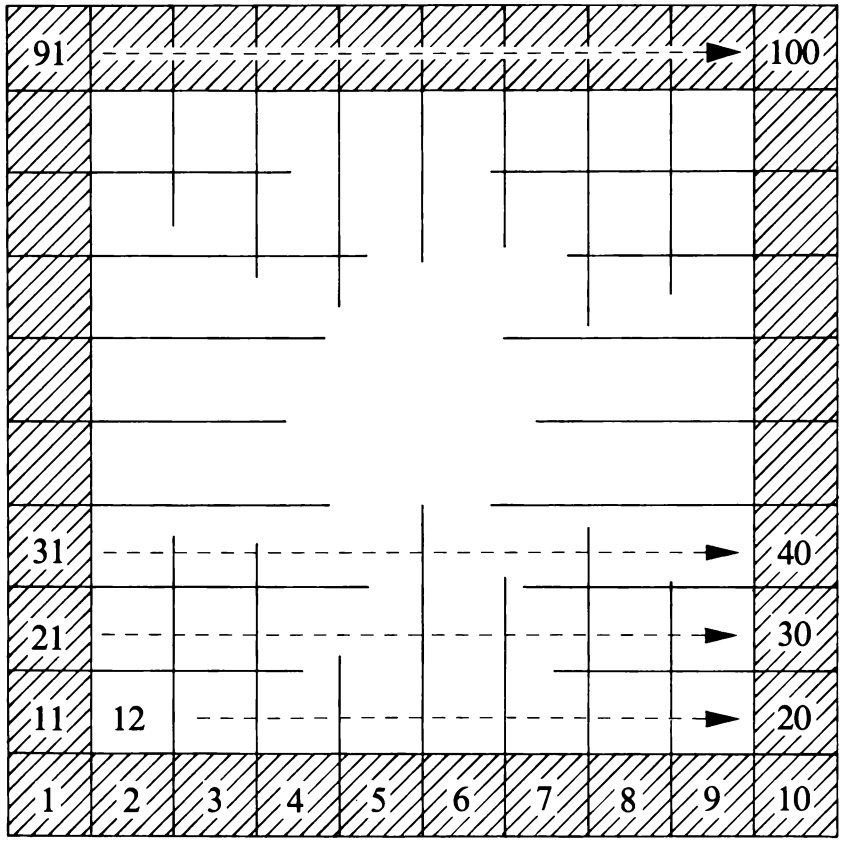
To complete the complex, we have to scatter a few walls at random throughout our dungeon, to make for a bit of variety in the game.

As the computer looks through array C, or, as we imagine it, our three levels of 100 cells each, on a 10×10 matrix, it is going to check the cells around each location, to see if any contain the value 999. As we know, this value is a marker which tells us that this particular cell, or location, is a no-go area. It may not seem important, at the moment, if we have two of these walls together, but let a cell be surrounded by walls and we have an impenetrable room — not much good in our present game! Although, you may want to write a routine to allow us, maybe to tempt a monster into such a room, and then imprison the beast. We may have, for instance, a Teleport Spell in order to escape ourselves.

The computer will also check, at the same time, that there are no diagonal walls across the dungeon, thus cutting us off from the rest of the cells.

```
1710 LET b=INT (RND*30)+30
```

Figure 6

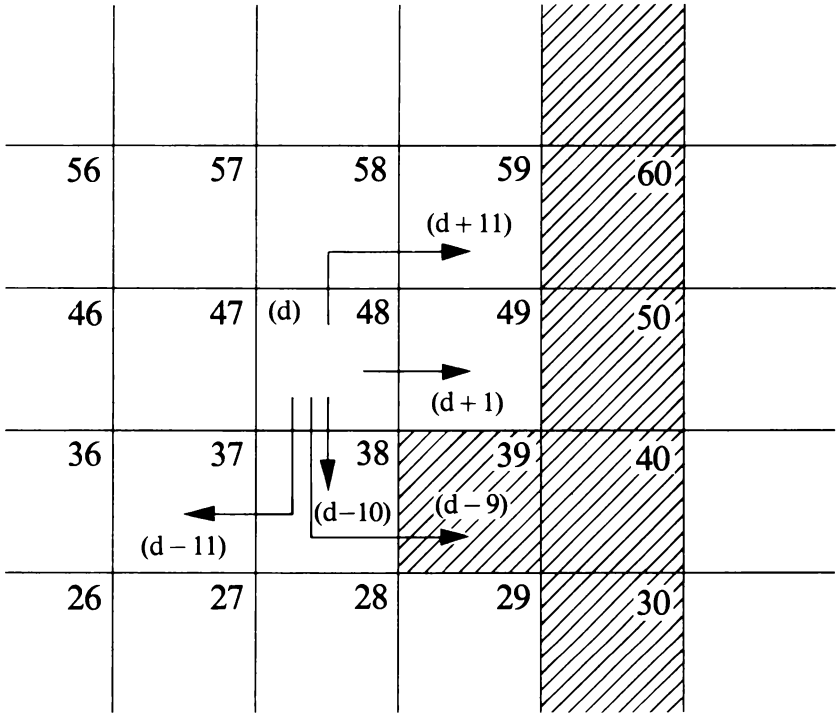


gives us a value for b, between 30 and 59. This is the range we require, of walls in each level, apart from the boundary walls.

```
1720 FOR a=1 TO b
1730 LET d=INT (RND*280)+10
1740 IF C(d)=999 THEN  GO TO 173
0
1750 IF C(d-10)=999 OR C(d+1)=99
9 OR C(d+11)=999 THEN  GO TO 173
0
1760 IF C(d-11)=999 OR C(d-9)=99
9 THEN  GO TO 1780
1770 LET C(d)=999
1780 NEXT a
```

This is the loop to create the walls. In line 1750 and 1760, the program is checking all the surrounding cells to the one chosen by 1730, to make sure none contain the value 999 — if one does, the program returns to find another value for d, and repeat the check. Look at **Figure 7** to see this in action. The illustration shows just one section of Level 1 (although it could just as easily be Level 2, or 3).

Figure 7



1730 gives us a value for d of between 10 and 290. These numbers should be familiar to us now, but to recap, we have walls between 1 and 10, and between 290 and 300, so there's no point trying to put a wall at these locations. In our example, the program is checking location 48 — that is, d has received the value of 48 from line 1730.

The program asks first of all if 48 has the value 999 (line 1740). It hasn't in our example, but if it did, the program would try again with a fresh value for d. Line 1750 asks if d − 10 (48 − 10 = 38) is a wall — it isn't, so line 1750 goes on to ask, does d + 1 (48 + 1 = 49) hold the value 999. Again the answer is no, so line 1750 finishes by asking if d + 11 (48 + 11 = 59), holds the value 999, and once again the answer is no. Again, if the answer to one of these

questions had been yes, the program would jump back to line 1730 to get a new value for d, and try once more.

In our example, however, the program goes on to line 1760 and asks if d – 11 (48 – 11 = 37) has the value 999, which, again, it doesn't. Does d – 9 (48 – 9 = 39) = 999? Zounds! it does, and now the program jumps to line 1780, which returns us to 1730 for a new value. If, on the other hand, no "wall" had been found surrounding C(48), that location would have been given the value 999, thus making it an impassable wall.

Now that all our cells are designated either walls, or rooms, we can go on to number those rooms.

```
1790 LET d=0
1800 FOR a=10 TO 290
1810 IF C(a)=999 THEN  GO TO 184
0
1820 LET d=d+1
1830 LET C(a)=d
1840 NEXT a
```

Now that our rooms are numbered, the computer will recognise a number between 10 and 290 as a clear area, and those locations with the value 999, as a wall, and thus impassable.

One last thing needs to be done before we can leave this phase of the program. How do we get between the various levels of the dungeon? Having gone to a lot of trouble to set up walls between the levels, we need a way of bypassing these walls as necessary.

Stairs have been a successful answer to this problem for several centuries, so let's put some of these in our complex.

We don't want to make it too easy for adventurers to find their way to another level, so let's put just one staircase on each level. For Level 1, we'll store the location of the staircase in variable SONE. We don't want our staircase in a wall, so we can forget locations 1 – 10 and 90 – 100.

```
2359 REM **********************
2360 REM SET UP STAIRS, HEALING
         WELL and FIRE PIT
2361 REM **********************
2370 LET SONE=INT (RND*90)+10
2380 IF C(SONE)=999 THEN  GO TO
2370
```

And so on for the other two levels:

```
2390 LET STWO=INT (RND*90)+110
2400 IF C(STWO)=999 THEN  GO TO
2390
2410 LET STHREE=INT (RND*90)+210
2420 IF C(STHREE)=999 THEN  GO T
O 2410
```

This finally completes the structural set-up of our dungeon complex. And although the routines were written with our final graphic adventure in mind, the techniques may just as well be used in your text adventure.

CHAPTER 9
All that Glisters...

On without pause to the good bit — populating our new dungeon with the enemy and the loot!

```
1849 REM **********************
1850 REM Populate dungeon with
         monsters
1851 REM **********************
1860 DIM M(290)
```

because we know the last 10 locations of array C are walls, and we want M to coincide with C. There will be a maximum of one monster to each room — monsters will not be able to roam around in this dungeon!

```
1870 FOR a=10 TO 290
1880 LET b=INT (RND*4)+1
1890 IF b<2 THEN GO TO 1910
1900 LET M(a)=INT (RND*15)+1
1910 NEXT a
```

Line 1870 also recognises that the last 10 locations are walls, but also that the 1 to 10 are also walls. Line 1900 gives a random choice of 15 monsters, while the other three lines give us a frequency of about one monster every other room.

```
1920 LET M(12)=0
```

Location number 12 in the array M is set at 0, that is, no monster will appear there. If you remember, the array M will coincide with the array C, which holds our network of rooms and walls, and line 1920 ensures that we don't start the game having to battle with a monster! (Why are we starting the game at C(12)?).

```
1930 LET a=INT (RND*80)+210
1940 IF C(a)=999 THEN GO TO 1930
1950 LET M(a)=16
```

There is one Lich in our complex. He is the dead Wizard, who we will eventually have to battle in the climax to the adventure. He is a very special monster, and as such, is to be found somewhere on the third and final level. Line 1940 ensures that he doesn't end up in a wall, while line 1950 gives him a special code of 16.

Now to the treasures, which are actually mostly weapons. These can be dealt with in exactly the same kind of routine.

```
1959 REM **********************
1960 REM Place treasures in
         dungeon
1961 REM **********************
1970 DIM T(290)
1980 FOR a=10 TO 290
1990 LET b=INT (RND*4)+1
2000 IF b<3 THEN GO TO 2020
2010 LET T(a)=INT (RND*15)+1
2020 NEXT a
```

Array T, which now holds all the information on the treasures to be found in the complex, is also congruent with arrays M and C. Thus, as the player moves about array C, which holds the physical set-up of the dungeon, the other two arrays can be very quickly matched against C to get information on monsters and treasures currently at that location in C.

One last array, S, remains to be DIMensioned.

```
2067 REM **********************
2068 REM STORAGE SPACE
2069 REM **********************
2070 DIM S(290)
```

Array S will hold all the information on treasures that we may drop as we move around the dungeon. As with all our other arrays, we only need to DIMension the array to 290 locations — the last 10 locations form a wall, in array C. This may look, on the face of it, rather wasteful of memory space, as, after all, we may only drop one or two treasures throughout the whole game. The routine to search through a string, every time we entered a room, just in order to check if we had stored a treasure previously, would slow up the program by an unacceptable degree. Keeping all our arrays congruent, as we have done, our program has only to look through the same location in each array, as we enter each room — a very quick and economical process.

Now we have a two-dimensional string M$, which will hold the data on

19 kinds of enemy (not always creatures — see the last three, which are very special spells you'll find used against you in the final stages). The longest name is 15 characters long.

```
2079 REM **********************
2080 REM READ IN MONSTERS
2081 REM **********************
2090 DIM M$(19,15)
2100 FOR a=1 TO 19: READ M$(a):
NEXT a
2110 DATA "Living Skeleton","Mum
my","Demon","Zombie","Fire Eleme
ntal","Vampire","Mind Vampire","
Wraith","Dragon","Werewolf","Cyc
lops","Sandman","Harpie","Giant
Serpent","Balrog","Lich","Lightn
ing bolt","Stone spell","Limbo s
pell"
```

T$ is DIMensioned in the same way, and holds information on the various types of treasure.

```
2119 REM **********************
2120 REM READ IN TREASURES
2121 REM **********************
2130 DIM T$(25,15)
2140 FOR a=1 TO 25: READ T$(a):
NEXT a
2150 DATA "SPADE","FIRE WHIP","S
WORD","SILVER SWORD","SILVER STA
FF","SAINTLY STAFF","TALISMAN","
CROSS","SHIELD","TORCH","INVISIB
LE CLOAK","CLUB","HOLY WATER","B
OW AND ARROWS","MAGIC SHIELD","E
MPTY BOTTLE","HEALING WATER","WI
ZARDS HAND","TELEPORT","FORCESHI
ELD","PSYCHIC SHIELD","LIGHTNING
 BOLT","STONE SPELL","LIMBO SPEL
L","STONE"
```

The information on treasures is placed in a string at this point, so that we may PRINT the information on-screen during the game.

CHAPTER 10
Graphic Violence

Now that the structure of the dungeon is set, we can turn to defining some graphics. You will find, if you take a quick look through the complete listing at the end of the book, that there are many graphics defined in the main body of the program (look at lines 2960 to 3650). Each time a monster, or treasure appears, the graphics for that particular object will be redefined as necessary.

There are several graphics, however, in *The Eye of the Star Warrior*, that remain the same throughout the game. A lot of memory is saved by defining these right at the start, before the main program is loaded.

```
  10 BORDER 0: INK 0: PAPER 0: C
LS : PRINT  INK 2; BRIGHT 1; FLA
SH 1;AT 10,0;"
                Adventure Loadin9:
 Do Not Stop   Tape.

                "
  15 REM ** SET GRAPHICS **
  20 FOR a=0 TO 7
  30 REM .. WIZARDS HAND
  35 READ b: POKE USR "a"+a,b
  40 READ b: POKE USR "b"+a,b
  45 REM ..MAP ROOM WITHOUT
           TREASURE
  50 READ b: POKE USR "c"+a,b
  55 REM ..MAP ROOM WITH
           TREASURE
  60 READ b: POKE USR "d"+a,b
  70 REM .. STONE WALL
  80 READ b: POKE USR "i"+a,b
  85 REM .. PLAYER
  90 READ b: POKE USR "j"+a,b
100 REM *STAIRS*
110 READ b: POKE USR "k"+a,b
120 REM *HEALING WELL*
130 READ b: POKE USR "l"+a,b
```

```
 140 REM *FIRE PIT*
 150 READ b: POKE USR "m"+a,b
 160 NEXT a
 170 DATA BIN 00001111,BIN 11111
110,BIN 11110111,BIN 11110111,BI
N 11011011,BIN 00111000,BIN 0,BI
N 0,BIN 11011001
 180 DATA BIN 00011111,BIN 11111
110,BIN 10000001,BIN 10000001,BI
N 0,BIN 00111000,BIN 00000011,BI
N 00011100,BIN 11001010
 190 DATA BIN 11111111,BIN 11100
000,BIN 10000001,BIN 10010101,BI
N 01100110,BIN 00010000,BIN 0000
0011,BIN 00111110,BIN 00000010
 200 DATA BIN 11111111,BIN 11110
000,BIN 10000001,BIN 10001001,BI
N 0,BIN 11111110,BIN 00001111,BI
N 01111110,BIN 10110000
 210 DATA BIN 11111111,BIN 11100
000,BIN 0,BIN 00010100,BIN 11011
011,BIN 00010000,BIN 00001111,BI
N 01111100,BIN 00110101
 220 DATA BIN 11111111,BIN 11110
000,BIN 10000001,BIN 10100001,BI
N 0,BIN 00101000,BIN 00111111,BI
N 00111000,BIN 10000000
 230 DATA BIN 11111111,BIN 11100
000,BIN 10000001,BIN 11000001,BI
N 01100110,BIN 01000100,BIN 0011
1111,BIN 00011000,BIN 01101101
 240 DATA BIN 00111111,BIN 11110
000,BIN 11110111,BIN 11110111,BI
N 0,BIN 10000010,BIN 11111111,BI
N 0,BIN 01101100
 260 LOAD "Prog"
 270 SAVE "adventure" LINE 10
```

Note that some of the BIN numbers contain a single 0. The computer recognises this as eight 0s! We have used BINary statements throughout the graphic-defining procedures, in preference to decimal numbers, which, of course, are just as valid. The BIN numbers may be a little longer, but, in

our opinion, are less prone to typing errors, and are easier to read. They also, if you use a little imagination, approximate the shape of the final graphic.

You will also see that the letters in quotes are in lower case. These letters — a,b,c,d,i,j,k,l and m — are to be typed in having first gone to GRAPHIC mode on the computer (press CAPS SHIFT and the 9 key together). In this mode, the machine automatically goes into upper case. Throughout the listing, we've kept to showing these letters, rather than the graphic symbol you'll see when the program is RUN, thus making it easier for you to see which key to press when typing the listing. As a convention, we've underlined any letter (always in quotes anyway), that needs to be entered using the GRAPHIC mode.

RUN the little graphics program, and you'll find that keys now print nice graphics.

Let's start using those graphics.

We'll get the status display box on to our screens first.

```
2490 FOR a=0 TO 12: PRINT PAPER
0; INK 1;AT a,0;"
                  ": NEXT a
```

This line prints a nice background to our Status Box.

```
2500 PRINT "_____________________
_____"
```

The line between quotes is achieved by SHIFTed "0".

```
2510 PRINT PAPER 1; INK 5;AT 0,1
9;"ADVENTURE"
2520 PRINT PAPER 1; INK 7;AT 4,1
9;"STRENGTH     "
2530 PRINT PAPER 1; INK 7;AT 6,2
2;"FLOOR     "
2540 PRINT PAPER 1; INK 7;AT 8,1
4;"STATUS MODE       "
2550 PRINT PAPER 1; INK 7;AT 10,
14;"COMBAT RATING     "
2560 PRINT PAPER 1; INK 7;AT 12,
21;"WOUNDS     "
```

These lines will print a nice colourful box at the right of the screen.

Now to drawing the graphics of the room. As we enter each room we are going to get the computer to paint a picture of the room, complete with monsters, treasures, and your little man.

The best way for us to do this is to DIMension a string, which can hold in its two dimensions a replica of each room, and which will PRINT out the information on the room as we enter.

```
2480 DIM X$(12,12)

2569 REM ********************
2570 REM SET UP AND PRINT
        GRAPHICS OF ROOM
2571 REM ********************
2580 LET X$(1,1 TO 12)="
     "
2590 LET X$(2,1 TO 12)=" iiiiiii
iii "
2600 FOR a=3 TO 10
2610 LET X$(a,1 TO 12)=" i
  i "
2620 NEXT a
2630 LET X$(11,1 TO 12)=" iiiiii
iiii "
2640 LET X$(12,1 TO 12)="
     "
```

You will see the X$ will print out four solid walls — not much use, as we don't want to spend the game cooped up in the first room we come to!

So the next job is to check all the locations around us to see which are walls, and which are rooms, into which we can move.

```
2650 IF C(LOCATION-10)<>999 THEN
  LET X$(2,6 TO 7)="  "
2660 IF C(LOCATION+10)<>999 THEN
  LET X$(11,6 TO 7)="  "
2670 IF C(LOCATION-1)<>999 THEN
 LET X$(6,2)=" "
2680 IF C(LOCATION-1)<>999 THEN
 LET X$(7,2)=" "
2690 IF C(LOCATION+1)<>999 THEN
 LET X$(6,11)=" "
2700 IF C(LOCATION+1)<>999 THEN
 LET X$(7,11)=" "
```

These lines look at the locations all round our present location. When the program finds a room, rather than a wall, it overprints the wall graphic currently on-screen with two spaces, thus giving a graphic representation of our exit point.

Finally, we can PRINT X$. Type

```
3830 FOR a=1 TO 12
3840 PRINT TAB 1;X$(a,1 TO 12)
3850 NEXT a
```

Before we get to printing our little man, we must define a few variables

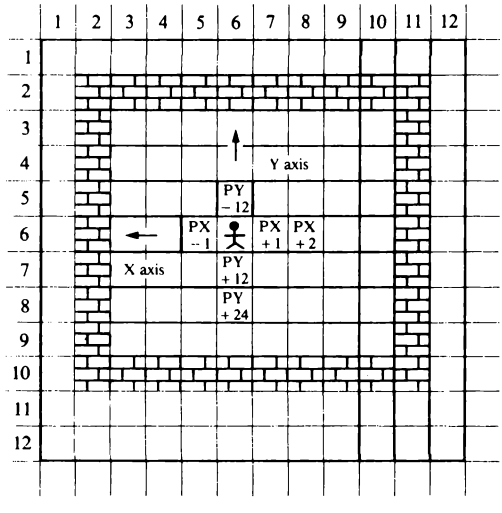
```
2470 LET FLOOR=1: LET LOCATION=1
2: LET PX=6: LET PY=6: LET STREN
GTH=100: LET COMBAT=0: LET SPELL
=0: LET PER=0: LET WOUNDS=0: LET
 SM=0: LET HELD=0: LET RET=0: LE
T WHAND=0: LET STONE=0
```

That's quite a long line, and introduces several things we haven't seen before — but many of them will probably be obvious. The variables we are interested in at the moment, are PX and PY, along with FLOOR and LOCATION. These last two are self-explanatory, but the other two are not immediately so. PX is the value of the Players X – (or horizontal) position, and PY is the Players Y – (or vertical) position. These are set at 6 and 6 at the very start of the game, and the player starts at LOCATION 12 (the first eleven are walls, you'll remember), on FLOOR (or Level) 1, and at co-ordinates 6X6. This works out as the middle of our 6X6 playing area. This is the simple (but clever!) reason for us putting the playing box at top left — all the information about the room will occur at the same co-ordinates in array C, array T, array M, and X$. This will save an awful lot of speed in the final game, as it will not be necessary for the program to check positions of all the objects at every move.

CHAPTER 11
Make your Move

Our little man is standing there, in the middle of Room 12, waiting for his adventure to begin. How do we move him? In line 2470, we initialised several variables, two of these being PX and PY. These are the players co-ordinates in the room, PX being the horizontal position, and PY being the vertical position. Their value at the start of the game is 6, and, if you look at **Figure 8**, you'll see that these values place us bang in the middle of the playing area.

Figure 8



To move to the left, we decrement the value of PX by one for each square, and to the right, we increment PX by one for each square. North and south movement is similar, but in these cases we increment and decrement by 12. At each stage, however, the values of BOTH PX and PY have to be stated. Thus the statement LET (PY + 12,PX) will move us south by one square (ie the vertical axis is changed by 12, while the horizontal axis remains as it was).

Type:

```
4400 LET TX=PX: LET TY=PY

4450 IF S$="5" THEN  LET TX=TX-1
4460 IF S$="6" THEN  LET TY=TY+1
4470 IF S$="7" THEN  LET TY=TY-1
4480 IF S$="8" THEN  LET TX=TX+1
```

We know that a wall is impenetrable — if we now attempted to move our man onto, or through a wall, the values of PX and PY would be lost forever. So our new variable TX and TY act as a suicide squad — if they are brought up short by a wall, PX and PY, which in the meantime have retained their position, are still held in X$. If a wall is not met, then PX and PY are updated to the position held by TX and TY, which then go on, if necessary to scout another position.

```
4550 IF X$(TY,TX)="i" THEN GO TO
 4040
```

sends us back to the Movement module at 4040 which we'll be typing in soon. If we find a break in the wall the next lines send us to the New Room module

```
4560 IF TY<2 OR TY>11 THEN GO TO
 4740
4570 IF TX<2 OR TX>11 THEN GO TO
 4740
```

And then lines 4660 and 4665 finally PRINT our player symbol and reset the variables for the next move.

```
4660 PRINT AT TY,TX;"j"
4665 LET PY=TY: LET PX=TX
4670 IF X$(PY,PX)<>"l" THEN GO T
O 4685
```

The discussion on movement is all very well in theory, but at the moment we can't actually talk to the computer to tell it what we want it to do. So let's now type in our Communication module:

```
 4030)LET a=0
4040 DIM B$(32)
4050 LET RET=0
4060 LET COMBAT=0
4070 LET k=0
4080 LET S$=INKEY$
4090 IF CODE S$=12 OR CODE S$=48
 THEN  GO TO 4230
4100 IF CODE S$=13 THEN  GO TO 4
270
4110 LET a=a+1
4120 IF a>=60/FLOOR THEN  GO TO
4930
4130 IF S$="" THEN  GO TO 4080
4140 IF S$="5" OR S$="6" OR S$="
7" OR S$="8" THEN  GO TO 4400
4150 IF k>=31 THEN  GO TO 4080
4160 LET k=k+1
4170 LET B$(k)=S$: LET B$(k+1)="
>"
4180 PRINT AT 21,0;B$: BEEP 0.1,
20
4190 LET a=a+1
4200 IF a>=60/FLOOR THEN  GO TO
4930
4210 IF INKEY$<>"" THEN  GO TO 4
190
4220 GO TO 4080
4230 IF k<1 THEN  GO TO 4080
4240 LET B$(k)=">": LET B$(k+1)=
""
4250 LET k=k-1
4260 GO TO 4180
4270 GO SUB 60
4280 IF B$( TO 4)="TAKE" THEN  G
O TO 5120
4290 IF B$(1)="I" THEN  GO TO 54
40
4300 IF B$( TO 4)="DROP" THEN  G
O TO 5590
4310 IF B$( TO 4)="WAIT" THEN  G
O TO 6070
4320 IF B$( TO 4)="LOOK" THEN  G
O TO 4850
```

```
4330 IF B$( TO 3)="DIG" THEN  GO
 TO 6290
4340 IF B$( TO 5)="DRINK" THEN
GO TO 6900
4350 IF B$( TO 2)="UP" OR B$( TO
 4)="DOWN" THEN  GO TO 6140
4355 IF B$( TO 3)="MAP" THEN  GO
 TO 8710
4360 IF B$( TO 8)="TELEPORT" THE
N  GO TO 6720
4365 IF B$( TO 4)="SAVE" THEN  G
O TO 8500
4370 PRINT  INK 5; BRIGHT 1;"I d
ont understand your command."
4380 GO TO 4040
```

B$ will hold our commands, and is DIMensioned to 32, because as you know, that is the width of the computer's screen. Typing in 33 characters would cause a "scroll?" message to appear. S$ we have encountered in our Movement module, and indeed, we now see, in line 4140, a command sending us to the Movement module when we press a cursor key. Line 4090 allows us to delete a character in one of two ways. CODE 12 is the code for DELETE, thus allowing us to use the CAPS SHIFT key together with the 0 key, in the usual way, while CODE 48 is the code for 0, and allows us to use just the 0 key to delete.

The variable "k" is a marker in B$, at the position of our current input, and (k + 1) will place a cursor arrow in the next position, prompting another input. The delete routine in lines 4090 sends us to line 4230, which merely replaces the last character with the cursor arrow, and sends "k" back to the previous position. Line 4180 PRINTs the B$ as it is entered, with the accompaniment of a little BEEP. Although this will slow the program down a little, and make input a little slower, that is its intention. When being chased by a monster, we don't want to make it too easy for the player to pick up treasure and run!

The module controls not only our input, and thus movement, commands, but also the speed of the monster. The variable "a" is set to 0 in line 4030, and then incremented by 1 each time a key press is made (either for movement or character input) — and in fact is incremented even when no key press is made. So while you're sitting there gazing at the beauty of the display, "a" is ticking away. Line 4120 takes the value of the variable FLOOR, set initially (in line 2470) to 1, and increasing by 1 as we move up level by level, and divides 60 by that value. The higher the level, the sooner the program is sent to line 4930, which is the Monster Move routine. Thus, if a monster is present in the current room, he will move inexorably toward

you. It will move even faster if you are inputting a text command — we don't want to make this game too easy for you.

There's one last line to be explained here. In line 4100, CODE 13 is the code for <ENTER>. Whenever your command (held now in B$) is ENTERed, the little routine at line 50 clears the bottom eight lines of the screen, B$, or your command along with them.

```
 49 REM ************************
 50 REM CLEAR BOTTOM 8 LINES
        OF SCREEN
 51 REM ************************
 60 GO SUB 530
 70 FOR d=14 TO 21
 80 PRINT AT d,0;"
                  "
 90 NEXT d
100 PRINT AT 13,0;"_"
110 RETURN
```

Line 60 sends us to the routine at 530, which will print our status on the status table, and which we will type in soon.

For the moment, however, let's look again at our variables in line 2470. SM, initially set at 0, is a counter for our strength, and is related to our movement.

This is our "prod", the "hurry up and do something" part of the game.

The player may wish to wander around the complex, admiring the view and keeping out of the way of monsters, but in our game, we want to keep him on the move. The way to do this is to decrement his strength at intervals. The only way to regain strength is to kill monsters, thus giving a pretty good incentive to the player to find a monster and start bashing. We could, if we felt really mean, reduce strength by 1 at each move — but let's be generous, and allow eight moves before the player's strength fails. Each room is 8×8, thus allowing the player to go from side to side before losing any strength.

SM is incremented by 1 each time a movement command is given, in line 4500

```
4500 LET SM=SM+1
4510 IF SM<=8 THEN GO TO 4550
```

As long as SM is less than 8, we jump on to 4550. But if SM should be equal to, or greater than 8, the variable STRENGTH, initially set to 100, is weakened by 1, plus the value of the variable WOUNDS. The program then goes back to the routine at 530, to update the status box display.

Now we can type in the routine, which starts at 530, and finishes at 750

```
  524 REM **********************
 525 REM PRINT STATUS TABLE
 526 REM **********************
 530 IF STRENGTH<=100 THEN  PRIN
T  PAPER 6; INK 0; BRIGHT 1;AT 2
,16;"    MISFIT     "
 540 IF STRENGTH>100 AND STRENGT
H<=250 THEN  PRINT  PAPER 5; INK
 0; BRIGHT 1;AT 2,16;"    JESTER
    "
 550 IF STRENGTH>250 AND STRENGT
H<=400 THEN  PRINT  PAPER 4; INK
 0; BRIGHT 1;AT 2,16;"     HERO
    "
 560 IF STRENGTH>400 AND STRENGT
H<=500 THEN  PRINT  PAPER 3; INK
 7; BRIGHT 1;AT 2,16;"   WARRIOR
    "
 570 IF STRENGTH>500 AND SPELL=0
 THEN  PRINT  PAPER 7; INK 1; BR
IGHT 1;AT 2,16;"  SUPER HERO  "
 580 IF STRENGTH<=500 THEN  GO T
O 610
 590 IF SPELL>0 AND SPELL<=3 THE
N  PRINT  PAPER 6; INK 0; BRIGHT
 1;AT 2,16;"    WIZARD     "
 600 IF SPELL>3 THEN  PRINT  PAP
ER 2; INK 1; BRIGHT 1; FLASH 1;A
T 2,16;" GRAND WIZARD "
 610 PRINT  PAPER 7;AT 4,28;"
 "
 620 IF STRENGTH>9999 THEN  LET
STRENGTH=9999
 630 PRINT  PAPER 7; INK 0;AT 4,
28;STRENGTH
 640 PRINT  PAPER 7; INK 0;AT 6,
29;FLOOR
 650 IF M(LOCATION)=0 THEN  PRIN
T  PAPER 4; INK 0; BRIGHT 1;AT 8
,26;" GREEN"
 660 IF COMBAT>0 THEN  GO TO 680
 670 IF M(LOCATION)<>0 THEN  PRI
NT  PAPER 6; INK 1; BRIGHT 1; FL
```

```
HSH 1;AT 8,26;"YELLOW"
 680 IF COMBAT>0 THEN  PRINT  IN
K 2; PAPER 7; BRIGHT 1; FLASH 1;
AT 8,26;"COMBAT"
 690 PRINT  PAPER 7; INK 0;AT 10
,28;"    "
 700 IF PER>0 THEN  PRINT  PAPER
 7; INK 0;AT 10,29;PER
 710 PRINT  PAPER 7; INK 0;AT 12
,29;WOUNDS
 720 IF STRENGTH<=0 THEN  GO TO
750
 730 PRINT AT 13,0;"_"
 740 RETURN
 750 GO SUB 70
```

Returning from this routine, SM is reset to 0, and the process starts again.

```
4040 LET SM=0
```

CHAPTER 12
Monster! Monster!

There is something missing at the moment from our program, and a pretty important something — we haven't yet provided graphics for monsters or treasures!

The first task, and I'm afraid that it's a long one, is to type out all the lines for defining these graphics. Starting at line 2970, we input the BIN numbers for the SPADE, to start with, and then on to all the other objects. We won't print all those lines here — turn to the complete listing at the back of the book, and type in lines 2960 to 3650.

You'll notice, from the REMarks before each graphic, that many of the monsters share a common top and (pardon the expression!) bottom. This not only saves memory, but also a lot of typing!

Now for another tiresome bit of typing — lines 119 to 520 contain the RESTORE routines for all the graphic data that we've just ploughed through.

At the back of the book, before the main listing, you'll find a few grids, which show you how each graphic character is built up. You may wish to alter these to suit your own idea of how the Monsters and treasure should look (although we think they're pretty good as they are!)

Each time we enter a new room, the program checks to see if any treasure, or a Monster is present. If the answer is yes, the graphic character, or characters, for it is READ, and PRINTED, as appropriate.

```
2740 LET a=T(LOCATION)
2750 IF a=0 OR a>=18 THEN GO TO
2790
2760 GO SUB 130
2770 FOR a=0 TO 7: READ b: POKE
USR "o"+a,b: NEXT a
```

If there is treasure here (in other words, if array T does indeed hold a treasure at this location), the local variable "a" is given the value of the treasure. Line 2760 then sends the program back to the RESTORE routine at 130, and then on to READ in the necessary graphic characters. Line 2750

forces the program on to line 2790, if the value of the treasure is 18 and above, or 0 — in both of these cases, no graphics are required.

The program then goes on to check the array S, which holds information on our "stored" treasures — that is, treasures that we may previously have dropped here. The routine is the same as for the one we have just typed in.

```
2790 LET a=S(LOCATION)
2800 IF a=0 THEN GO TO 2830
2810 GO SUB 130
2820 FOR a=0 TO 7: READ b: POKE
USR "P"+a,b: NEXT a
```

And we can do exactly the same for the monster graphics:

```
2840 LET a=M(LOCATION)
2850 IF a=0 THEN GO TO 3660
2860 GO SUB 280
2870 FOR a=0 TO 7
2880 READ b: POKE USR "q"+a,b
2890 READ b: POKE USR "U"+a,b
2900 NEXT a
```

There is one difference here — the monster graphics are defined from four character squares, so another set of lines is needed to build the extra characters':

```
2910 LET a=M(LOCATION): GO SUB 0
430
2920 FOR a=0 TO 7
2930 READ b: POKE USR "r"+a,b
2940 READ b: POKE USR "t"+a,b
2950 NEXT a
```

And now, to place the treasure (both found and stored) in a random position in the newly-entered room:

```
3660 LET WSC=0:LET WFC=0:LET MC=
0:LET SY=1:LET SX=1:LET WX=1:LET
 WY=1:LET MY=0:LET MX=0:LET PER=
0:LET BY=0:LET BX=0
3667 REM **********************
3668 REM PLACE OBJECTS IN NEW
        ROOM.
3669 REM **********************
3670 IF T(LOCATION)<1 OR T(LOCAT
```

```
ION)>17 THEN  GO TO 3710
3680 LET WX=INT (RND*5)+4: LET W
Y=INT (RND*5)+4
3690 LET X$(WY,WX)="O"
3700 LET WFC=W(T(LOCATION))
3710 IF S(LOCATION)<1 THEN  GO T
O 3760
3720 LET SX=INT (RND*5)+4: LET S
Y=INT (RND*5)+4
3730 IF SX=WX AND SY=WY THEN  GO
 TO 3720
3740 LET X$(SY,SX)="P"
3750 LET WSC=W(S(LOCATION))
3760 IF T(LOCATION)<>0 THEN  GO
TO 3800
3770 IF T(LOCATION)=0 THEN  LET
BY=INT (RND*5)+4
3780 IF T(LOCATION)=0 THEN  LET
BX=INT (RND*5)+4
3790 IF X$(BY,BX)<>" " THEN  GO
TO 3760
```

WX and WY are the co-ordinates of the treasure (or Weapons), and are set, in line 3680, to at least 4X and 4Y, to leave room for the Healing Well and Fire Pit, which we will come to shortly. 3690 puts the co-ordinates into X$, and 3700 gives WFC (Found Weapon Colour), the value of the treasure. SX and SY are the co-ordinates for the Stored Treasure, and in line 3720, these are given a random value, with line 3730 trapping any attempt to give them the same value as WX and WY, which would result in the Found Treasure being overprinted by the Stored Treasure. WSC is Stored Weapon Colour. 3770 and 3780 give us the co-ordinates for Buried Treasure (BX and BY), although, being buried, they obviously don't need colours. We shall deal more fully with Buried Treasure later.

And finally, we can PRINT the treasures:

```
3870 PRINT INK WFC; BRIGHT 1;AT
WY,WX;"O"
3880 PRINT INK WSC; BRIGHT 1;AT
SY,SX;"P"
```

That sorts out the treasure — but there may also be a monster in the room, as we enter, so let's see him!

```
3940 LET MY=6: LET MX=6

3980 PRINT INK MC;BRIGHT 1;AT MY
,MX;"qu";AT MY+1,MX;"rt"
```

These two lines start the monster off in the middle of the room, and his colour (MC) will be taken from lines 2290 to 2310. To tidy everything up, we'll add three lines that send the program to three subroutines. These are at 60, which we've typed in already, and which clears the bottom of the screen of any text, at 530, and again we've typed this in, and 790. We will cover this later, and it PRINTS the information on what we have found in the room as we enter it.

So now we have our little yellow man at the dungeon's entrance, and our huge, flaming magenta Balrog at centre stage. What now — do we just stand looking at each other? Well, yes — at least until we put in a routine to get the monster moving! Let's do that now!

```
 4930 LET TY=MY:LET TX=MX
4935 LET a=0
4940 IF PY<TY THEN  LET TY=TY-1
4950 IF PY>TY THEN  LET TY=TY+1
4960 IF PX>TX THEN  LET TX=TX+1
4970 IF PX<TX THEN  LET TX=TX-1
4980 IF TY<3 OR TY>9 OR TX<3 OR
TX>9 THEN  GO TO 4080
4990 PRINT AT MY,MX;X$(MY,MX);AT
 MY,MX+1;X$(MY,MX+1)
5000 PRINT AT MY+1,MX;X$(MY+1,MX
);AT MY+1,MX+1;X$(MY+1,MX+1)
5010 PRINT  INK WFC;AT WY,WX;X$(
WY,WX)
5020 PRINT  INK WSC;AT SY,SX;X$(
SY,SX)
5030 IF HW=LOCATION THEN  PRINT
 INK 5;AT 3,4;"l"
5040 IF FP=LOCATION THEN  PRINT
 INK 2; PAPER 6; BRIGHT 1; FLASH
 1;AT 3,4;"m"
5050 IF T(LOCATION)=0 AND WHAND=
1 THEN  PRINT  INK 4; BRIGHT 1;A
T BY,BX-2;"ab"
5060 PRINT AT PY,PX;"j"
5070 PRINT  INK P(M(LOCATION));
```

```
BRIGHT 1;AT TY,TX;"qu";AT TY+1,T
X;"rt"
5080 LET MY=TY: LET MX=TX
5090 IF MY=PY AND MX=PX OR MY=PY
 AND MX+1=PX OR MY+1=PY AND MX=P
X OR MY+1=PY AND MX+1=PX THEN  G
O TO 7590
5100 GO TO 4080
```

In line 4930, we see the same logic at work as in the player's movement — that is, TY and TX are the forward scouts for MY and MX, the monster's co-ordinates. 4940 sees a difference, however. As you'll remember, PX and PY are the player's co-ordinates, and lines 4940–4970 check the values of PX and PY, relative to MX and MY, and constantly update the values of MX and MY. In this way, the monster is constantly moving towards the player's position. Line 4200, which we covered a while ago, sends the program to this routine every so often, and, as we saw then, more often with each successive level.

Lines 4990 and 5000 print out X$ after the monster has passed that location, thus restoring any locations that have been overprinted by the monster graphics, as it passed by.

If the monster is lured into a corner, line 4980 picks this up and returns us to the Input Command module, where the whole procedure starts again.

Assuming, though, that the Monster is still on the move, line 5070 PRINTS its graphics at the co-ordinates found by TY and TX. These variables are then replaced by the main variables, MX and MY.

Lines 5010, 5020 and 5060 repair any graphics left by the passage of the player (PX and PY, you'll remember), by rePRINTing any treasure graphics.

Then, the good bit, which we've all been waiting for! Line 5090 checks to see if any of the monster's character squares coincides with those of the player's (at PX and PY). If not, the program jumps back to line 4080, our Input Command routine.

However, if any character position coincides, combat is begun, and the Combat routine is found at line 7590.

Actually, this routine is the Combat Defence routine — there is also a Combat Attack routine which we shall go into later, but for now, the monster has attacked you, and you must defend.

Before the player can attack a monster, or defend himself, he must pick up some weapons. In the next chapter, we go treasure-hunting. But don't worry, we'll be back to monster-bashing soon enough!

CHAPTER 13
Trusty Weapons

From nearly the very beginning of the game, you will come across treasure. You'll also come across lots of monsters, and you will have to avoid them until you've collected a good arsenal.

When the adventurer first enters a room, a list of its contents is PRINTed, after the initial graphics have been displayed. This list includes information on what treasure is here, as well as which monster is guarding the treasure. The player will also be told whether there are stairs, or the Fiery Pit, or the Healing Well. If this is the cave containing the Wizard's Hand, then the Hand is transferred to the player's possession, where it will remain until the Lich is finally vanquished.

```
 779 REM **********************
 780 REM PRINT OUT CONTENTS OF
       ROOM
 781 REM **********************
 790 LET r=T(LOCATION)
 800 IF r=0 OR r=99 THEN  GO TO
830
 810 IF r=18 THEN  GO TO 980
 820 GO TO 840
 830 IF S(LOCATION)=0 THEN  GO T
O 920
 840 PRINT "You have found: ";
 850 IF r=0 OR r=99 THEN  GO TO
870
 860 PRINT T$(r)
 870 LET r=S(LOCATION)
 880 IF r=0 THEN  GO TO 920
 890 IF T(LOCATION)=0 OR T(LOCAT
ION)=99 THEN  GO TO 910
 900 PRINT "and ";
 910 PRINT T$(r)
 920 IF COMBAT=3 THEN  GO TO 940
 925 IF M(LOCATION)<>0 THEN  PRI
NT "The room is guarded by a"
```

```
 930 IF M(LOCATION)<>0 THEN  PRI
NT M$(M(LOCATION))
 940 IF X$(3,10)="k" THEN  PRINT
 "You see stairs in the corner."
 950 IF X$(3,4)="l" THEN  PRINT
"You see a healing well."
 960 IF X$(3,4)="m" THEN  PRINT
"You see a fire pit."
 970 RETURN
 980 PRINT "You have found the g
hostly hand of the good Wizard.
It is yours until you find the e
ye."
 990 FOR r=1 TO 12
1000 IF H(r)=0 THEN  GO TO 1020
1010 NEXT r
1020 LET H(r)=18
1030 LET WHAND=1
1040 LET T(LOCATION)=99
1050 GO TO 920
```

If the player likes what he finds, and can get past the monster to get it, he will want to TAKE the treasure, and this is where the TAKE module comes in:

```
5109 REM **********************
5110 REM TAKE OBJECT
5111 REM **********************
5120 IF X$(PY,PX)<>"l" THEN  GO
TO 5200
5130 FOR r=1 TO 12
5140 IF H(r)=16 THEN  GO TO 5170
5150 NEXT r
5160 PRINT "You need an empty bo
ttle to     collect water from t
he healing  well.": GO TO 4040
5170 LET H(r)=17
5180 PRINT "You have collected a
 bottle fullof healing water."
5190 GO TO 4040
5200 IF X$(PY,PX)="o" OR X$(PY,P
X)="P" THEN  GO TO 5230
5210 PRINT "I see nothing to Pic
k uP."
```

```
5220 GO TO 4040
5230 IF HELD<5 THEN  GO TO 5260
5240 PRINT "Sorry; you cannot ca
rry anymore treasures."
5250 GO TO 4040
5260 FOR r=1 TO 12
5270 IF H(r)=0 THEN  GO TO 5300
5280 NEXT r
5290 GO TO 5240
5300 IF X$(PY,PX)="o" THEN  LET
H(r)=T(LOCATION)
5310 IF X$(PY,PX)="P" THEN  LET
H(r)=S(LOCATION)
5320 PRINT "You have picked uP:"
5330 IF X$(PY,PX)="o" THEN  PRIN
T T$(T(LOCATION))
5340 IF X$(PY,PX)="P" THEN  PRIN
T T$(S(LOCATION))
5350 LET HELD=HELD+1
5360 IF X$(PY,PX)="o" THEN  LET
T(LOCATION)=99
5370 IF X$(PY,PX)="P" THEN  LET
S(LOCATION)=0
5380 LET X$(PY,PX)=" "
5390 LET r=INT (RND*290)+1
5400 IF C(r)=999 OR T(r)=18 THEN
  GO TO 5390
5410 LET T(r)=INT (RND*15)+1
5420 GO TO 4040
```

Line 5120 checks to see if we are at present standing on the Healing Well. Line 5140 makes sure that we have a bottle, before we attempt to pick up any Healing Water. If H(r) = 16, the Empty Bottle, you are allowed to pick up a bottle full of water — if not, line 5160 PRINTs a message and returns the program to the Input Command module, to await another command.

Line 5200 ensures that the player is actually standing over treasure — if not, a message is PRINTed again, and the program returned to the Command Input module.

Line 5230 checks the number of treasures carried, which must not be more than 5.

Lines 5300 to 5380 check what the treasure is that you have moved on to, and PRINTs the information, finally, putting a space in X$ where the treasure was.

If the treasures held are up to the maximum, the player may like to drop something in order that he may pick up something more useful to him. Before doing this, why not let him see an inventory of what he is carrying?

First of all, let's put the command in the Command Input module:

```
4290 IF B$(1)="I" THEN  GO TO 54
40
```

And then the Invent module:

```
5429 REM ******
5430 REM INVENT
5431 REM ******
5440 GO SUB 60
5450 LET d=13: LET r=0
5460 FOR b=1 TO 12
5470 IF H(b)=0 THEN  GO TO 5520
5480 LET d=d+1
5490 IF d=20 THEN  LET r=17
5500 IF d=20 THEN  LET d=14
5510 PRINT AT d,r;T$(H(b))
5520 NEXT b
5530 IF COMBAT>0 THEN  GO TO 555
0
5540 GO TO 4040
5550 PAUSE 500
5560 IF COMBAT=1 THEN  GO TO 714
0
5570 IF COMBAT=2 THEN  GO TO 761
0
5575 IF COMBAT=3 THEN  GO TO 809
0
```

5440 sends the program back to clear the bottom eight lines in preparation for the inventory list. Lines 5470 to 5520 PRINT a list of the treasures you currently hold, and then the program goes back to Command Input module for your next command. If you are in the middle of a combat sequence, however (checked in line 5530), line 5550 counts down from 100 to 1 and then takes the program on to the Defence or Attack module, whichever is appropriate.

And now the player may, if he wishes drop an item of treasure.

```
5579 REM ****
5580 REM DROP
5581 REM ****
5590 LET RET=1
5600 IF PY=2 OR PY=11 OR PX=2 OR
 PX=11 THEN  GO TO 5620
5610 GO TO 5640
5620 PRINT "You cannot drop obje
cts in the  doorway."
5630 GO TO 4040
5640 LET B$(k+1)=""
5650 LET S$=B$(6 TO 20)
5660 FOR r=1 TO 22
5670 IF S$=T$(r) THEN  GO TO 571
0
5680 NEXT r
5690 PRINT "I don't understand w
hat you wantme to drop."
5700 GO TO 4040
5710 IF r<18 THEN  GO TO 5780
5720 IF r<>18 OR r<>25 THEN  GO
TO 5760
5730 PRINT "You cannot drop the"
5740 PRINT T$(r)
5750 GO TO 4040
5760 PRINT "You cannot drop some
thing that  exists in your memor
y."
5770 GO TO 4040
5780 FOR b=1 TO 12
5790 IF H(b)=r THEN  GO TO 5830
5800 NEXT b
5810 PRINT "You cannot drop some
thing you   do not carry."
5820 GO TO 4040
5830 LET H(b)=0
5840 IF S(LOCATION)=0 OR T(LOCAT
ION)=99 THEN  GO TO 5870
5850 PRINT "Invisible Imps carry
 off the    treasure you have ju
st dropped."
5860 GO TO 6040
5870 IF S(LOCATION)=0 THEN  GO T
O 5890
```

```
5880 GO TO 5960
5890 LET S(LOCATION)=r
5900 GO SUB 2790
5910 LET SX=INT (RND*5)+4: LET S
Y=INT (RND*5)+4
5920 IF X$(SY,SX)<>" " THEN  GO
TO 5910
5930 LET X$(SY,SX)="p": LET WSC=
W(r)
5940 PRINT  INK WSC; BRIGHT 1;AT
 SY,SX;X$(SY,SX)
5950 GO TO 6020
5960 LET T(LOCATION)=r
5970 GO SUB 2740
5980 LET WY=INT (RND*5)+4: LET W
X=INT (RND*5)+4
5990 IF X$(WY,WX)<>" " THEN  GO
TO 5980
6000 LET X$(WY,WX)="o": LET WFC=
W(r)
6010 PRINT  INK WFC; BRIGHT 1;AT
 WY,WX;X$(WY,WX)
6020 PRINT AT PY,PX;"j"
6030 IF M(LOCATION)<>0 THEN  PRI
NT  INK P(M(LOCATION)); BRIGHT 1
;AT MY,MX;"qu";AT MY+1,MX;"rt"
6040 LET HELD=HELD-1
6050 GO TO 4030
```

Line 5600 ensures that nothing is dropped in the doorway, and lines 5640 to 5680 check the player's input with the list of treasure in T$, to make sure that a valid input has been made. Only one treasure may be dropped in a room, and if the player attempts to drop a second, no warning is given — line 5850 ensures that he is properly penalised. Most of the other lines in the module are concerned with putting the Stored Treasure at a random position in the room, putting that position in X$ for future reference, and PRINTing the treasure. Other graphics in the room, such as the player and monster, both of whom will probably be moving again, are updated at the same time. We've used this method of updating before, and it saves a lot of time, compared with going to an update module every time it was needed.

Now that our player is fully equipped with appropriate weapons, he can face the monster. There are two modules that we can turn to during the Combat Phase — the first we need will be the Defence routine. Type this in now, and we'll discuss the details once you've completed this chore.

```
7569 REM **********************
7590 REM COMBAT DEFENCE
7591 REM **********************
7600 IF M(LOCATION)=16 THEN  GO
TO 8060
7610 GO SUB 60
7620 LET COMBAT=2
7630 IF M(LOCATION)=7 THEN  GO S
UB 1410
7640 PRINT "You are under attack
 choose yourdefence."
7650 IF INKEY$<>"" THEN  GO TO 7
650
7660 INPUT S$
7670 IF S$="I" OR S$="INVENT" TH
EN  GO TO 5440
7680 IF S$="LOOK" THEN  GO TO 48
50
7690 GO SUB 60
7700 PRINT "Choose your defence.
"
7710 IF LEN S$>32 THEN  GO TO 77
30
7720 PRINT  INK 5; BRIGHT 1;S$
7730 GO SUB 1070
7740 IF r>0 THEN  GO TO 7770
7750 GO SUB 1470
7760 GO TO 7640
7770 GO SUB 1150
7780 IF TRE=1 THEN  GO TO 7820
7790 PRINT "You cant use that."
7800 IF COMBAT=3 THEN  GO TO 803
0
7810 GO TO 7980
7820 IF S$="TELEPORT" THEN  GO T
O 6720
7830 LET WEAPON=r
7840 LET STRENGTH=STRENGTH-(1+WO
UNDS)
7850 IF r>18 THEN  LET STRENGTH=
STRENGTH-49
7860 GO SUB 1320
7870 IF r>0 THEN  GO TO 7920
7880 PRINT "Your defence did not
```

```
 Protect you"
7890 GO SUB 1520
7900 IF COMBAT=3 THEN  GO TO 803
0
7910 GO TO 7980
7920 PRINT "You defended well."
7930 GO SUB 1520
7940 FOR r=1 TO 50: NEXT r
7950 IF COMBAT=3 THEN  GO TO 812
0
7960 GO TO 7140
```

Line 7600 sends us to the Lich-fighting module, if he is the monster in the present room. If he isn't (and there's only one of him anyway), we go on to 7610 which returns us to the display-clearing sequence. 7620 updates the variable COMBAT, while 7630 checks for the presence of the Mind Vampire, who rates a special subroutine of his own, at 1410. Once asked to choose a weapon to defend, your answer is placed in S$, which is then compared, in lines 7670 and 7680. These two lines check to see if you have asked for an inventory of weapons currently held, or a sneaky look at what is in the room. We give this option to the player, because in the panic of the situation, he will probably have completely forgotten what he is holding, and what he is actually fighting in this room!

The inventory module we have covered in the section on dropping weapons — now type in the LOOK module:

```
4839 REM **********************
4840 REM LOOK COMMAND
4841 REM **********************
4850 GO SUB 60
4860 GO SUB 790
4870 IF COMBAT>0 THEN GO TO 4890
4880 GO TO 4040
4890 IF COMBAT=3 THEN PRINT "The
 Lich is attacking you with a";M
$(M(LOCATION))
4900 PAUSE 300
4905 IF COMBAT=1 THEN GO TO 7140
4910 IF COMBAT=2 THEN GO TO 7610
4915 GO TO 8090
```

The Look module is a fairly simple one — merely updating the Contents display according to what is in the room you are currently in. We will be typing this routine in shortly.

In the meantime, the player is still in the Defence mode, and after visiting the Inventory module, or the Look module, is returned to line 7700. Line 7720 PRINTS the choice on-screen as the player makes it, and then goes to the module at 1070.

```
1059 REM **********************
1060 REM CHANGE TREASURE INPUT
         INTO NUMBERS
1061 REM **********************
1070 DIM V$(LEN S$)
1080 FOR r=1 TO 25
1090 LET V$=T$(r)
1100 IF V$=S$ THEN RETURN
1110 NEXT r
1120 LET r=0
1130 RETURN
```

This module takes the length of the input the player has made in choosing his weapon, and compares it to the treasures held in T$. If the check is valid, that is, if the input is indeed the name of a treasure, then "r" is given the value of that treasure and the program returned to our Defence module, at line 7740. If an invalid choice has been made, that is if the player should accidentally type "sausages" as a choice of weapon, the program goes to line 1470 to say that it does not understand, then clears the bottom eight lines, ready for another input. If the program accepts the input as valid, it then carries on to check that the weapon asked for, is actually carried by the player.

```
1139 REM **********************
1140 REM CHECK FOR TREASURE
1141 REM **********************
1150 LET TRE=0
1160 FOR b=1 TO 12
1170 IF H(b)=r THEN GO TO 1200
1180 NEXT b
1190 RETURN
1200 LET TRE=1
1210 RETURN
```

TRE is set to 0 to begin with, and then line 1160 sets up a loop of twelve passes (12 being the maximum number of treasures that the player can carry, at any time). If the weapon is, indeed, presently in the player's possession, we go on to line 1200, which resets TRE to the value 1.

If the weapon is not carried by the player, line 7790 informs him of the fact. Otherwise, the program goes on to line 7830, where WEAPON is given the value of "r", found in our validation module at 1059.

STRENGTH is decremented by a certain amount whenever a treasure, or weapon is used, and by a lot more when a spell is used (line 7850).

If the adventurer has chosen unwisely, line 7800 sends him to be killed at 8030, and on to the Restart module at 6850.

7860 is really the core of this module. Each monster has a Combat rating, as we've seen, and only certain weapons may be used against it. The little module at 1320 checks the effect of the weapon that the player has chosen, to defend himself against the monster:

```
1309 REM **********************
1310 REM CHECK YOUR DEFENCE
1311 REM **********************
1320 LET B$=D$(M(LOCATION))
1330 FOR a=1 TO 18 STEP 2
1340 LET S$=B$(a TO a+1)
1350 IF S$="  " THEN  GO TO 1380
1360 IF VAL S$=r THEN  RETURN
1370 NEXT a
1380 LET r=0
1390 RETURN
```

When the program returns to the Defence module, it will have a value for "r" — if it is more than 0, that is, if the player's choice of weapon is found to be effective against this particular monster, then we go on to the message at line 7920. If, however, the weapon was unwisely chosen, we go to the Damage Weapon module. Even if the player defended well, there is a danger of the weapon being damaged, but, at least it will have done its job.

```
1509 REM **********************
1510 REM DAMAGE WEAPONS
1511 REM **********************
1520 IF WEAPON=13 OR WEAPON=17 T
HEN  GO TO 1570
1525 IF WEAPON>16 THEN  RETURN
1530 LET r=INT (RND*90)+1
1540 IF r+(WOUNDS*10)+(COMBAT*10
)<90 THEN  RETURN
1550 PRINT "You have damaged the
"
1560 PRINT T$(WEAPON)
1565 PAUSE 100
```

```
1570 FOR a=1 TO 12
1580 IF H(a)=WEAPON THEN  GO TO
1600
1590 NEXT a
1600 IF WEAPON=13 OR WEAPON=17 T
HEN  GO TO 1625
1605 LET H(a)=0
1610 LET HELD=HELD-1
1620 RETURN
1625 LET H(a)=16
1630 RETURN
```

This is the Damage to Weapon module. The first line, 1520, checks the present value of WEAPON. The value of finding any other value for WEAPON, the program then gives "r" a random value between 1 and 90. This module is used in both Defence and Attack modes, and you can see from line 1540, that if the player is currently defending, with no wounds, that he will be unlucky to suffer damage to his defending weapon, and will usually be returned, weapon intact, to the Combat mode.

If, however, he is wounded, and particularly when in the Attack mode, the player stands a good chance of breaking his weapon. Line 1540 will send the program on to PRINT the information, and then to decrement HELD, the variable for the number of weapons currently held by the player.

Back to the Combat Defence module we go, where line 7940 causes the program to pause for a while, as "r" counts from 1 to 50. Then line 7950 checks the value of COMBAT — if the value is 3, this means that you are defending against the Lich. If the player has chosen the wrong weapon or spell, he is, unfortunately, dead!

Otherwise, we are returned to the Attack module, to allow the player his turn to bash the monster.

First of all, the bottom eight lines of the screen are cleared, as usual, and the information that the player is now in Attack mode flashed on the Status box (by the Display routine at 530).

```
 7140 LET COMBAT=1:GO SUB 50
7150 PRINT "Choose a weapon and
attack."
7160 IF INKEY$<>"" THEN  GO TO 7
160
7170 INPUT S$
7180 IF S$="W" OR S$="S" OR S$="
N"  OR S$="E" THEN  GO TO 4400
7190 IF S$="I" OR S$="INVENT" TH
```

```
EN  GO TO 3440
7200 IF S$="LOOK" THEN  GO TO 48
30
7210 IF LEN S$>32 THEN  GO TO 72
30
7220 PRINT  INK 5; BRIGHT 1;S$
7230 GO SUB 1070
```

In the Attack mode, the player has the choice of moving away from the monster, and 7180 checks S$ to see if this option has been taken. If it has, we are sent back to the Move Player module at 4410:

```
4420 IF S$="S" THEN  LET S$="6"
4430 IF S$="N" THEN  LET S$="7"
4440 IF S$="E" THEN  LET S$="8"
4450 IF S$="5" THEN  LET TX=TX-1
4460 IF S$="6" THEN  LET TY=TY+1
4470 IF S$="7" THEN  LET TY=TY-1
4480 IF S$="8" THEN  LET TX=TX+1
```

The player may also "LOOK" if he wishes.

The next lines are very similar to lines 7710–7790 in the Combat Defence module, in checking the validity of the player's input of attack weapon.

```
 7210 IF LEN S$>32THEN GO TO 7230
7220 PRINT  INK 5; BRIGHT 1;S$
7230 GO SUB 1070
7240 IF r>0 THEN  GO TO 7270
7250 GO SUB 1470
7260 GO TO 7150
7270 GO SUB 1150
7280 IF TRE=1 THEN  GO TO 7310
7290 PRINT "You don't have that
weapon."
7300 GO TO 7610
```

If the player has mis-typed (a very costly mistake in this game!), or asked for a weapon that he doesn't possess (ditto!), the Attack Phase finishes. Having made the right choice, however, the Attack Phase continues with:

```
7310 IF S$="TELEPORT" THEN  GO T
O 6720
7320 LET STRENGTH=STRENGTH-(WOUN
DS+1)
7330 IF r>18 THEN  LET STRENGTH=
STRENGTH-49
7340 LET WEAPON=r
```

```
7350 GO SUB 1230
7360 IF r>0 THEN  GO TO 7410
7370 PRINT "Your weaPon is usele
ss a9ainst"
7380 PRINT "the ";M$(M(LOCATION)
);"."
7390 GO SUB 1520
7400 GO TO 7610
```

which are all similar in effect to lines 7820 — 7890 in the Combat Defence module, with the exception of line 7350, which goes to the Attack Weapons Effect, rather than the Defence Weapons Effect.

Now, line 7360 causes the program to jump to 7410, which gives a random result of the attack, and compares it to PER, the monster's percentage chance of vanquishing the player.

```
 7410 IF INT (RND*100)+1<PERTHEN
GO TO 7470
```

If the value of PER is sufficiently high, the player merely wounds the monster, and the program increments PER accordingly (because the player is a little weaker, after such sterling effort, that the monster's chances become a bit better), or, if PER is sufficiently small, tells the player that he has killed the monster.

```
7420 PRINT "You have wounded the
"
7430 PRINT M$(M(LOCATION))
7440 LET PER=PER+10
7450 IF PER>100 THEN  LET PER=10
0
7455 GO SUB 1520
```

and then returns to the Combat Defence routine:

```
7460 GO TO 7610
```

or goes to 7470, if the monster at the present location is the Lich, from where it goes to the Lich's death routine at 8210.

```
7470 IF M(LOCATION)=15 THEN  GO
TO 8210
```

If, however, the random number at 7410 is greater than the monster's percentage chance of beating the player, the program carries on to:

```
7475 PRINT "You have destroyed t
he"
```

```
7480 PRINT M$(M(LOCATION))
7485 GO SUB 1520
7490 LET M(LOCATION)=0
7500 PRINT AT MY,MX;X$(MY,MX);X$
(MY,MX+1)
7510 PRINT AT MY+1,MX;X$(MY+1,MX
);X$(MY+1,MX+1)
7520 PRINT AT PY,PX;"j"
```

The variable M(LOCATION) — ie this room, is set for 0, and X$ is rePRINTed, as is the player graphic at PY,PX.

```
7530 PAUSE 100
```

gives a short wait, and then all the variables are reset, and the screen cleared:

```
7540 LET STRENGTH=STRENGTH+(100-
PER)
7550 LET COMBAT=0: LET PER=0
7560 GO SUB 0050
7570 LET MY=0: LET MX=0: LET P=0
7580 GO TO 4030
```

And finally, the program returns us to the Input Command module, to await the player's next command.

Having discussed the two Combat modules, which are the heart of this whole program, we must type in three routines that are very important to both of the Combat modules.

The first one is the routine to read the values for the player's Attack weapons. Only certain weapons are of any use against particular monsters — for instance, a Spade can do a lot of damage to a Living Skeleton, but an Invisible Cloak won't damage him at all! Similarly, in defending himself against an attacking monster, the player has to choose the correct weapon.

The second module checks the values of the player's defence weapons, which depends on the particular monster he is defending against.

The third module gives a combat rating to each monster, which is used to calculate that monster's chances of wounding or killing the player.

So let's type in the first module, which is our Attack Weapon Data:

```
 2159 REM ********************
2160 REM READ IN ATTACK WEAPONS
         FOR EACH MONSTER
2161 REM ********************
```

```
2170 DIM A$(16,20)
2180 FOR a=1 TO 16: READ A$(a):
NEXT a
2190 DATA "0103040506071222224","
02071022232 4","06071324","020710
2324","06071324","020405061013322
2324","020405061013222324","0607
24","0304222324","020406071022223
24","030414222324","010506071213
222324","02030414222324","020304
10222324","03040714222324","2324
```

A$ is DIMensioned to 16,20. This allows for data on 16 monsters, the maximum amount of data being 20 characters long. Notice that most of the 0s are leading zeros — that is, the program will only accept two-digit numbers.

The Defence Weapon Data module is similar:

```
 2199 REM **********************
2200 REM READ IN DEFENCE
        WEAPONS AGAINST EACH
        MONSTER
2201 REM **********************
2210 DIM D$(19,18)
2220 FOR a=1 TO 19: READ D$(a):
NEXT a
2230 DATA "09111519","0209101115
19","0708151920","02070910111519
","071319","020706091011131519",
"020708091011131519","0708151920
","09111519","020709101113 1519",
 020910111519","091113151920","0
20910111519","020910111519","070
91519","192021","1520","21","21"
```

And finally, the Combat Rating module assigns values to array V, so that each monster has a unique rating to start combat — that value will always be the same, at least to start with. This is the reason for these values being placed in an array, rather than a string, as with the weapon lists, from which different information will have to be taken at different times.

```
2239 REM **********************
2240 REM COMBAT RATINGS
2241 REM **********************
2250 DIM V(16)
```

```
2260 FOR a=1 TO 16: READ V(a): N
EXT a
2270 DATA 70,70,20,40,70,50,50,5
0,20,50,20,50,60,20,30,20
```

Having spoken of normal monsters, there are a couple which are out of the ordinary, and both have their own module.

The first of these is the Mind Vampire. His unfortunate effect is to drain your mind of any spells that you may have found — very nasty, particularly as you cannot fight the Lich without a spell!

```
 1410 GO SUB 60
1420 FOR r=1 TO 12
1430 IF H(r)>18 THEN  LET H(r)=0
1440 NEXT r
1445 LET SPELL=0
1450 PRINT "Your mind is being d
rained."
1460 RETURN
1470 PRINT  INK 2; BRIGHT 1; FLA
SH 1;"        I dont understand.

1480 FOR a=1 TO 25: BEEP 0.1,10:
 BEEP 0.1,30: BEEP 0.1,20: NEXT
a
1490 GO SUB 60
1500 RETURN
```

After clearing the screen, the module cycles through array H, which is the treasure held by the player. When and if the program finds any treasure with a value greater than 18 (that is, a spell), it replaces it with the value 0, and then tells the player what has happened.

This, of course, is very tame, when compared to what the Lich will do to the player!

If the player stumbles across the Lich in his wanderings, the program jumps to the module at 8060

```
 8060 LET LICH=INT (RND*200)+500
8070 LET COMBAT=3
8080 LET SP=INT (RND*3)+17
8090 GO SUB 60
8100 PRINT "The Lich attacks. Pr
otect your- self from a ";M$(sp)
8105 LET M(LOCATION)=sp
8110 GO TO 7650
```

```
8120 LET LICH=LICH-50
8130 IF LICH>=50 THEN  GO TO 808
0
8140 LET M(LOCATION)=16
8150 LET PER=20
8160 PRINT "The Lich has exhaust
ed its      magic Powers."
8170 PAUSE 200
8180 GO TO 7140
```

The first line sets the Lich's strength to a random, but very high, figure, while 8080 gives him one of three spells to throw at the player in the first phase of combat. Each time the player defends well, the Lich loses 50 points of strength, until, when the variable LICH is less than 50, the player finally gets a chance to attack.

Should the player defeat the Lich, the program goes to the module at 8210.

```
8199 REM **********************
8200 REM LICH'S DEATH
8201 REM **********************
8210 PRINT "The Lich is dead. Yo
u have won  the magic stone- Eye
 of the StarWarrior. Now you mus
t DESTROY   it. The hand of the
wizard      vanishes. Its job is
 over.
8220 FOR r=1 TO 12
8230 IF H(r)=18 THEN  LET H(r)=2
5
8240 NEXT r
8250 LET WHAND=0: LET STONE=1
8260 GO TO 7490
```

8210 announces the fact, and 8220, 8230 and 8240 search through array H for the Wizard's Hand, replacing it with the Stone. WHAND is now 0, and STONE is set to 1. Line 8180 sends the program back to tidy up the screen.

CHAPTER 14
"I Spell a Rat. . . ."

How to fight the Lich? Obviously none of your ordinary weapons like the Spade, or even the Invisible Cloak are going to help the player in this situation!

Well we provided some special weapons for the confrontation, and these are spells, of one sort or another. They are not, of course, just lying about waiting to be picked up. Let's see how the player can get to them.

When seeding the dungeon complex at the start of the game, the program makes roughly 1 in every 4 treasures, a buried treasure, with a value of 0.

Because this treasure is buried, the player needs some kind of aid in finding, and recovering it. The next module will introduce this help to the program.

```
2030 REM * ADD WIZARDS HAND *
2040 LET a=INT (RND*70)+20
2050 IF C(a)=999 THEN  GO TO 204
0
2060 LET T(a)=18
```

The Wizard's Hand is placed in the first level, somewhere around the middle of the floor, between locations 20 and 70.

WHAND is set, in line 2470, to 0. As we found when typing in the Contents module, WHAND is then set to 1, when and if the player finds the Hand in his wanderings.

The Hand will stay with the player until he finally vanquishes the Lich, at the climax of the game. Whenever a room is entered that contains buried treasure, the Hand will glow and point, in a very friendly gesture towards the location.

And how does the player get to the treasure once it has been very kindly pointed out to him? Well, one method of getting buried treasure is to dig for it, and you will probably have noticed, in the list of treasure data, a Spade — in fact, it is number one on the list. Once one of these has been found, and TAKEn, the player may use it to dig up the treasure indicated by the Wizard's Hand.

```
 6290)FOR r=1TO 12
6300 IF H(r)=1 THEN  GO TO 6340
6310 NEXT r
6320 PRINT "You will need a spad
e."
6330 GO TO 4040
6340 IF BY=PY AND BX=PX THEN  GO
 TO 6370
6350 PRINT "There is nothing bur
ied here."
6360 GO TO 4040
6370 IF WHAND=1 THEN  GO TO 6400
6380 PRINT "You won't find burie
d treasure  without the Wizards
hand."
6390 GO TO 4040
```

These first few lines of the Dig module ensure that the player is carrying a Spade, with which to dig, and the Wizard's Hand, which will show him where to dig.

```
6400 LET r=INT (RND*4)+1
6410 PRINT AT BY,BX-2;X$(BY,BX-2
);X$(BY,BX-1)
6420 GO SUB 60
6430 IF r>2 THEN  LET b=INT (RND
*6)+19
6440 IF r<=2 THEN  LET b=INT (RN
D*15)+1
6450 IF b>18 THEN  GO TO 6550
6460 LET T(LOCATION)=b
6470 PRINT "You have found: ";T$
(b)
6480 LET WY=BY: LET WX=BX
6490 LET BY=0: LET BX=0
6500 LET RET=1
6510 GO SUB 2740
```

6400 gives "r" a random value between 1 and 4, and, after PRINTing the Hand in line 6410, and clearing the screen with subroutine 60, the program moves on to lines 6430 and 6440. These two lines take the value of "r" and depending on whether it is less than, or greater than 2, give the player a random treasure. This could be either an "ordinary" treasure, or a spell. The value of the treasure ("b") is put into T(LOCATION) for future

reference, and BY and BX are reset to 0, the co-ordinates of the Buried Treasure now becoming the co-ordinates for the usual Found Treasure. This means that the treasure will be able to be shown on-screen, as the player hasn't picked it up as yet. 6470 informs the player of what he has found.

Now we can place the graphic character for the newly-dug treasure into X$ and find its colour, in WFC.

```
6520 LET X$(WY,WX)="o"
6530 LET WFC=W(T(LOCATION))
6540 GO TO 4040
```

and the return to the Input Command module, in order to get the player's TAKE command. If he decides to TAKE, then HELD is updated, and the TAKE routine at 5120 carried out.

Should the treasure turn out to be a spell (that is, one with a value greater than 18), the player is told what spell he may now use (and he doesn't have to TAKE a spell — it is transferred direct to his memory).

```
6550 PRINT "You can now use a:"
6560 PRINT T$(b)
```

You'll remember that when we were setting up the dungeon, way back at the beginning of this section of the book, we gave the value 999 to any location that we wanted to designate as a wall, or no-go area.

We can use the same idea to denote a location that doesn't, at present, contain a treasure (as for instance when treasure has just been picked up). So that we don't confuse matters, let's give this condition the value 99, which is as easy to see in the listing as 999.

```
6570 LET T(LOCATION)=99
6580 LET d=INT (RND*290)+1
6590 IF T(d)=0 OR C(d)=999 THEN
 GO TO 6580
6600 LET T(d)=0
```

Thus, 6570 sets the value of T(LOCATION), or the present room, to 0, which means that there is now no treasure of the found variety — the player can still store a treasure here. 6560 finds a random value for "d" from 1 to 290 (the number of rooms in the dungeon). 6590 ensures that a room already containing treasure (with the value of 0), or a wall (with the value of 999), is not seeded with a treasure, while 6600 gives the treasure the value of 0, that is, makes it a Buried Treasure. In this dungeon, there happen to be lots of Invisible Imps (well, of course you haven't seen them!) which will

rush about planting treasure as the player picks up. They will plant the same kind of treasure that is picked up. And it may be in the same room that the player is currently looting, but won't be PRINTed on-screen until the room is entered once more.

```
6610 LET BY=0: LET BX=0
6620 FOR r=1 TO 12
6630 IF H(r)=b THEN  GO TO 4040
6640 NEXT r
6650 FOR r=1 TO 12
6660 IF H(r)=0 THEN  GO TO 6680
6670 NEXT r
6680 LET H(r)=b
6690 LET SPELL=SPELL+1
6700 GO TO 4040
```

BY and BX, being redundant now, are set to 0 at line 6610. Lines 6620–6640, and 6650–6670 are two little loops that check the value in array H (the player's present treasure-holding), against what the player has picked up. A spell is held in the adventurer's memory, so, unless his mind is wiped clean by a mind vampire, he need only pick up one spell, of a particular sort. The lines here check to see if the player is carrying a spell of the sort he has picked up, and either return him to the Input Command module, or increment SPELL. In the case of ordinary treasure, he is anyway returned to the Input Command module, which allows him to TAKE the treasure, should he wish.

Like the Mind Vampire, a very special monster, there is a very special spell — and that is the Teleport Spell. As its name suggests, it can instantly whisk you away from the scene of a battle to an unspecified location within the dungeon. You'll see, from line 7310 and 7820 that the program provides for the player selecting this option in both the defence and attack modes. The module at 6720 first of all — yes, you guessed it, clears the bottom eight lines of the screen.

```
6709 REM ********
6710 REM TELEPORT
6711 REM ********
6720 GO SUB 60
6730 FOR d=1 TO 12
6740 IF H(d)=19 THEN  GO TO 6780
6750 NEXT d
6760 PRINT "You don't know that
spell."
6770 GO TO 4040
```

6730 and 6740 check through the list of treasures currently held by the player in array H and informs him of his mistake in trying to use a weapon that he doesn't possess, and then (at 6770), returns him to the Input Command module for another go. If the check finds a valid input, however, at 6740, we go to 6780 and on:

```
6780 LET LOCATION=INT (RND*290)+
1
6790 IF C(LOCATION)=999 THEN  GO
 TO 6780
6800 LET PY=3: LET PX=3:
6810 LET STRENGTH=STRENGTH-50
6820 GO SUB 530
6830 GO TO 2580
```

6780 selects a random location from one of the 290 possible (although line 6790 ensures that the player isn't deposited in a brick wall), and the co-ordinates for the player's position in the new location are set at PX,PY = 3 that is, in a corner on level 3. This could prove rather embarassing, as a monster, if one is present, will hurry towards him. 6810 decrements the player's STRENGTH by 50 (Teleport doesn't come cheap), and 6820 updates the screen display. 6830, finally, PRINTs the graphics of the new room, and starts everything over again.

We have now covered the main modules of the game, from set-up, through creation of monsters and treasures, and their graphics and colours. Combat and movement have been looked at also, and now there is a bit of tidying-up to do. In the next chapter we'll look at the details to make our game even better.

CHAPTER 15
Nice and Tidy...

Unfortunately, we haven't finished typing yet, as there are a few modules to be added to our (already very large) program.

Apples and pears....

We have talked about the different levels of the dungeon — three in fact — and we have created stairs between them.

Now to PRINT the graphics in the appropriate place:

```
2710 IF SONE=LOCATION OR STWO=LO
CATION OR STHREE=LOCATION THEN
LET X$(3,10)="k"
```

Now we should allow our player to be able to climb and descend, using those stairs.

```
6129 REM **********************
6130 REM CLIMB STAIRS
6131 REM **********************
6140 IF X$(PY,PX)="k" THEN  GO T
O 6170
6150 PRINT "You can't do that --
 you must   find the stairway."
6160 GO TO 4040
6170 IF B$( TO 2)="UP" AND LOCAT
ION=STHREE THEN  GO TO 6260
6180 IF B$( TO 4)="DOWN" AND LOC
ATION=SONE THEN  GO TO 6260
6190 IF B$( TO 2)="UP" AND LOCAT
ION=STWO THEN  LET LOCATION=STHR
EE
6200 IF B$( TO 2)="UP" AND LOCAT
ION=SONE THEN  LET LOCATION=STWO
6210 IF B$( TO 4)="DOWN" AND LOC
ATION=STWO THEN  LET LOCATION=SO
NE
```

```
 6220 IF B$( TO 4)="DOWN" AND LOC
ATION=STHREE THEN  LET LOCATION=
STWO
6230 IF B$( TO 2)="UP" THEN  LET
 FLOOR=FLOOR+1
6240 IF B$( TO 4)="DOWN" THEN  L
ET FLOOR=FLOOR-1
6250 GO TO 2580
6260 PRINT "You can't go that wa
y."
6270 GO TO 4040
```

The first line of the module, 6140, checks that there are indeed stairs to be climbed at the present location. If not, the player is returned to the Input Command module, to give another command. If there is a staircase, then the player's directions are analysed in the following lines. Silly inputs, such as "DOWN" at Level 1 are trapped, and the resulting location of the player worked out. The variable FLOOR is also amended as necessary, the new room's graphics set up and PRINTed, and finally, the program returns once more to the Input Command module.

Have a drink on me....

Having placed the dread Lich on the top, and most dangerous level, we could put a little spice into the program by adding something different from all the other treasures, one on each of the other levels. If you plan to write your own adventures, this is a good technique to keep the player's interest.

```
 2430>LET HW=INT (RND*90)+1
2440 IF C(HW)=999 THEN  GO TO 24
30
```

These two lines place the Healing Well somewhere within the first level, along the way ensuring that the Healing Well doesn't end up in the wall.

```
2720 IF HW=LOCATIONTHEN LET X$(3
,4)="1"
```

Places HW, the variable for the Healing Well, in X$, ready to be PRINTed as necessary by line 3890:

```
3690 IF HW=LOCATION THEN  PRINT
INK 5;AT 3,4;"l"
```

The player can use the Healing Well in one of two ways — by moving on to it, he may first of all find his wounds automatically healed.

```
4670 IF X$(PY,PX)<>"l" THEN  GO
TO 4685
4675 LET WOUNDS=0
4680 GO SUB 530
```

And the program returns to 530, in order to update the screen display.

Otherwise the player may, after finding a Bottle, fill it from the Well, and carry it to the upper levels — only one drink is allowed, however, though the player may return for more water at any time.

"You're getting warmer..."

On the second Level, we can place the Fire Pit. This is where the player will bring the Stone (The Eye of the Star Warrior), once he has finally managed to defeat the Lich. This is set up and PRINTed in exactly the same way as the Healing Well.

```
2450 LET FP=INT (RND*90)+100
2460 IF C(FP)=999 THEN  GO TO 24
50
```

2450 sets it in the second level, with 2460 checking that it will not be placed in a wall.

The Player gains the Stone as soon as he kills the Lich, and we have typed in the Lich's death module already — the transfer of the Stone occurs at line 8250, which sets the value of variable STONE from 0 to 1. The same line changes the value of WHAND from 1 to 0, thus effecting the changeover. Now the player has to negotiate his way through the third level again, back to the second level, where the Fire Pit is situated. Moving onto the Fire Pit's position will end the game, and the player has won.

```
4685 IF STONE=1 AND X$(PY,PX)="m
" THEN  GO TO 7070
```

More room, more room...

And, while we're speaking of moving through the levels, how does the player actually move from room to room? This is accomplished at lines 4740–4830.

```
 4729 REM **********************
4730 REM MOVE TO NEW ROOM
4731 REM **********************
4740 IF S$="5" THEN  LET LOCATIO
N=LOCATION-1
=11
4760 IF S$="6" THEN  LET LOCATIO
N=LOCATION+10
4770 IF S$="6" THEN  LET PY=2
4780 IF S$="7" THEN  LET LOCATIO
N=LOCATION-10
4790 IF S$="7" THEN  LET PY=11
4800 IF S$="8" THEN  LET LOCATIO
N=LOCATION+1
4810 IF S$="8" THEN  LET PX=2
4820 FOR r=1 TO 11: PRINT AT r,0
;"                ": NEXT r
4830 GO TO 2580
```

These lines should be fairly obvious to us by now. Lines 4560 and 4570 have sent us here, the program having picked up the fact that the player is standing in the doorway of the previous room. The first set of lines, 4740–4810, reading the cursor keys, and updating to the required location. Line 4820 prints lines of blanks, thus clearing the previous room display,and the program returns to line 2580, where the new room's graphics are PRINTed.

Where am I?...

All this wandering about, and picking up treasure and bashing monsters is likely to disorientate the player. Even though each room is numbered on the screen display, the player is likely to forget exactly where he is, and where he has left useful treasure. So let's provide a little map for him. Type in the Map module:

```
8699 REM ***
8700 REM MAP
8701 REM ***
8710 IF FLOOR=1 THEN  LET b=11
8720 IF FLOOR=2 THEN  LET b=111
8730 IF FLOOR=3 THEN  LET b=211
8735 LET f=14: LET g=1
8740 FOR a=b TO 9+b
8750 LET e=0: LET d=6: LET g=g+1
8760 IF C(a)=999 THEN  GO TO 885
0
```

```
8770 IF S(a)<>0 THEN  LET d=4
8780 IF M(a)<>0 THEN  LET e=1
8790 LET S$="c"
8800 IF S(a)>0 OR T(a)<>99 THEN
 LET S$="T"
8805 PRINT  INK d; BRIGHT 1; FLA
SH e;AT f,9;S$
8810 IF a=LOCATION THEN  PRINT A
T f,9;"P"
8820 NEXT a
8830 IF f=21 THEN  GO TO 8860
8835 LET f=f+1: LET g=1: LET b=b
+10
8840 GO TO 8740
8850 PRINT AT f,9;"i"
8855 GO TO 8820
8860 PRINT AT 17,17;"Press ENTER
"
8865 IF INKEY$="" THEN  GO TO 88
65
8870 GO SUB 60
8875 GO TO 4030
```

The Map takes the form of a plan view of the level that the player is currently on, with each room, and the walls, printed. The monsters are shown by flashing squares, and treasure is also shown. Thus a yellow square with the little treasure symbol denotes a cave with treasure, while a flashing green square will denote a square with a monster present. An empty cave is also shown — and, of course, all the combinations, as well as the player's position. The module doesn't really need line-by-line explanation, and, in fact, it might be a good end-of-book exercise for the reader to work through it and follow the logic, armed with what he has learnt of all the variables used in the game.

Er, hold on a minute...

During combat, the phone might ring, as it usually does at a critical moment, so we could give the player a WAIT command, which will suspend the action until he returns.

```
6059 REM ****
6060 REM WAIT
6061 REM ****
6070 GO SUB 60
6080 PRINT' INK 2; BRIGHT 1; FLA
```

```
SH 1;" WAITING "; INK 5; FLASH 0
;"Press Enter to continue"
6090 IF INKEY$<>"" THEN  GO TO 6
090
6100 IF INKEY$="" THEN  GO TO 61
00
6110 GO SUB 60
6120 GO TO 4040
```

It pays to save...

In the same vein, and if the player should have to leave the game overnight, we can now type in the last major module, and that is the Save Game routine:

```
8489 REM **********************
8490 REM SAVE GAME ROUTINE
8491 REM **********************
8500 SAVE "adventure" LINE 8600
8510 SAVE "graphics" CODE 65368,
1968
8520 SAVE "screen"SCREEN$
8530 GO TO 4040
8600 LOAD "graphics"CODE 65368,1
68
8610 LOAD "screen"SCREEN$
8620 BORDER 0: PAPER 0: INK 6
8630 POKE 23658,8
8640 GO TO 4040
```

We are sent here from the Input Command module, and the routine SAVEs, on tape, the state of the game as reached. All the screen graphics are also SAVEd, so that when the game is next LOADed, it will restart at the same point.

Line 8630 is a POKE to ensure that all input is in upper-case (or capitals).

And, finally, one or two little modules that we must type in to round off the program. The first is the Victory, or End of Game, module, which the program will come to if the player has successfully completed his mission, and hurled the Stone into the Fiery Pit:

```
7059 REM **********************
7060 REM END OF GAME
7061 REM **********************
7070 PRINT  INK 2; PAPER 0; BRIG
HT 1; FLASH 1;AT PY,PX;" "
7080 GO SUB 60
```

```
7090 PRINT "The eye is destroyed
 and the    land will remain fre
e of evil."
7100 GO TO 6850
```

The routine at 6850 is the Restart module:

```
6839 REM *******
6840 REM RESTART
6841 REM *******
6850 PRINT "Press ENTER to start
 new game."
6860 IF INKEY$<>"" THEN  GO TO 6
860
6870 IF INKEY$="" THEN  GO TO 68
70
6880 RUN
```

And now we just start the whole game off with:

```
   5 BORDER 0: PAPER 0: INK 6: C
LS : PRINT AT 11,0;" Press ENTER
 to set up dungeon. "
  10 PAUSE 0
  15 RANDOMIZE
  20 PRINT  INK 2; BRIGHT 1; FLA
SH 1;AT 10,0;"
                PLEASE WAIT.SETTI
NG UP DUNGEON
                "
  30 POKE 23658,8
  40 GO TO 1640
```

and two lines at the very end to SAVE this whole listing:

```
8989 REM ********************
8990 REM SAVE PROGRAM TO AUTO
         RUN
8991 REM ********************
9000 SAVE "Prog" LINE 5
9010 STOP
```

Endplay

And now we've at last finished typing, you'll be pleased to read! The game, we think, is pretty good, and has been thoroughly tested — the listing, as you can see, has been taken direct from the program listing. We could say that we have included a deliberate bug, and ask you to spot it — which should cover any strange anomalies that game authors always miss, but which seem to turn up eventually. But any mistakes we have made are purely accidental! Let us know of any you find, or, indeed, improvements you make to the game (we're already planning *The Eye of the Star Warrior, Part Two*!)

And improving upon the game is really the idea behind this book — although this game is great fun to play (and very addictive too), it is really only a framework around which you can build your own game. You may, for instance, be a dedicated *D&D* fan, and, although this game has its basis in the *D&D*-style of combat, a non-graphic game could use many of the modules in this book. Thus, more detail or more monsters, may be included. Use the modules in any way you like.

16K-ers are, I'm afraid, not going to be able to enjoy this game as written — the program as it stands absolutely crams the 48K memory! However, a deal of ingenuity could be pressed into service, and *The Eye of the Star Warrior* converted to the smaller memory. For instance, being content with one level would save a little memory. And leaving the Lich sequence out would save more — in this case, the player's task could be to kill as many monsters as possible. A score routine could then be introduced. Graphics could still be used, but the monsters would have to be just one or two, single-character symbols.

If you can run *The Eye of the Star Warrior* as LISTed — have lots of fun!

We haven't included every single line of the listing in the text — some lines are repeated in several modules, which saves returning to subroutines to PRINT graphics and so on, when updating displays. Thus, you will quite often find lines similar to 6000–6030 reappearing, in the complete listing. This particular little routine will rePRINT the screen graphics as they are corrupted by the passing of the monster, or the player's symbol.

APPENDIX A
Defined graphics

Here you will find a selection of the graphics characters used in *The Eye of the Star Warrior* — feel free to alter them as you wish.

Figure 1.1 Fire Elemental

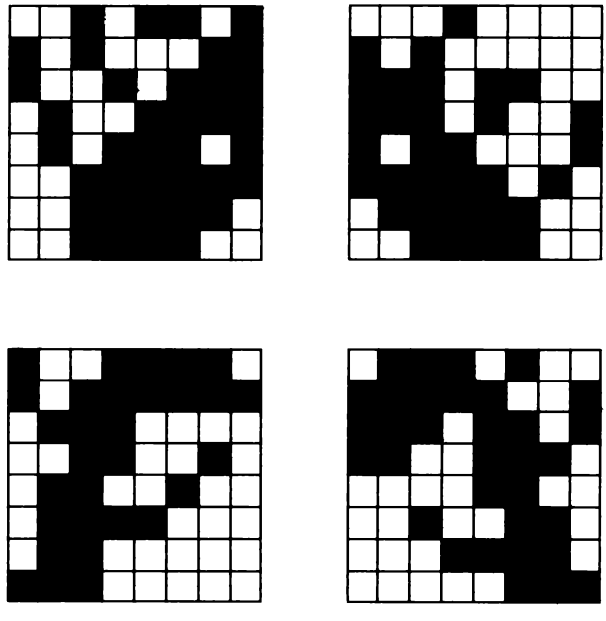


Figure 1.2 Giant Serpent

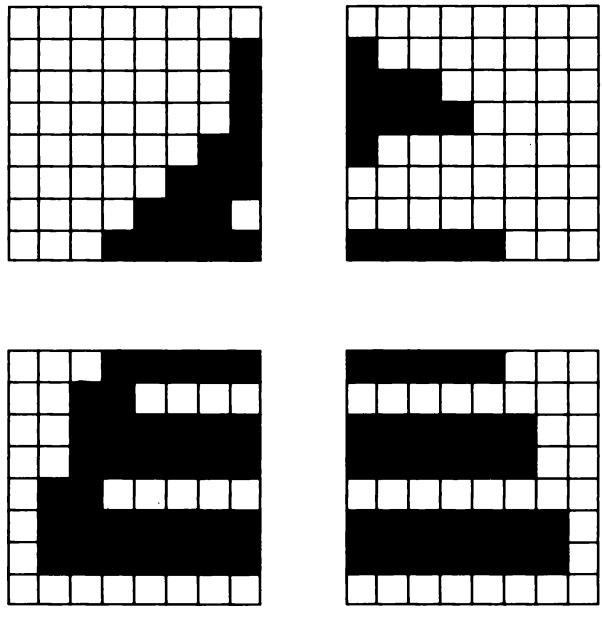


Figure 1.3 Sandman

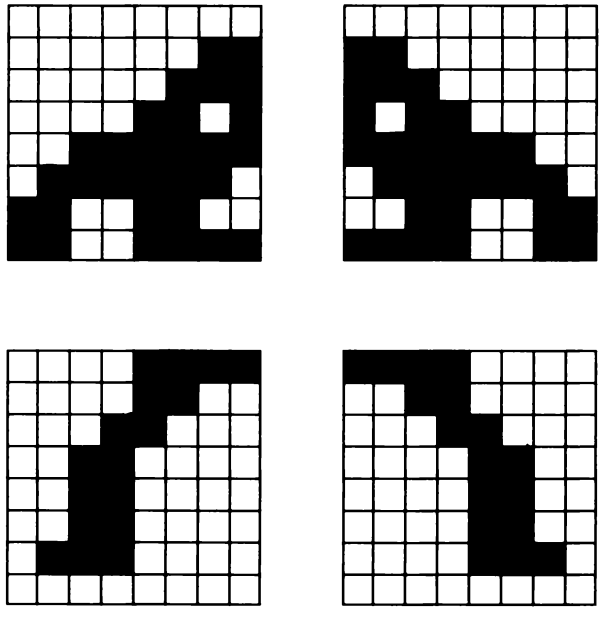


Figure 1.4 Dragon

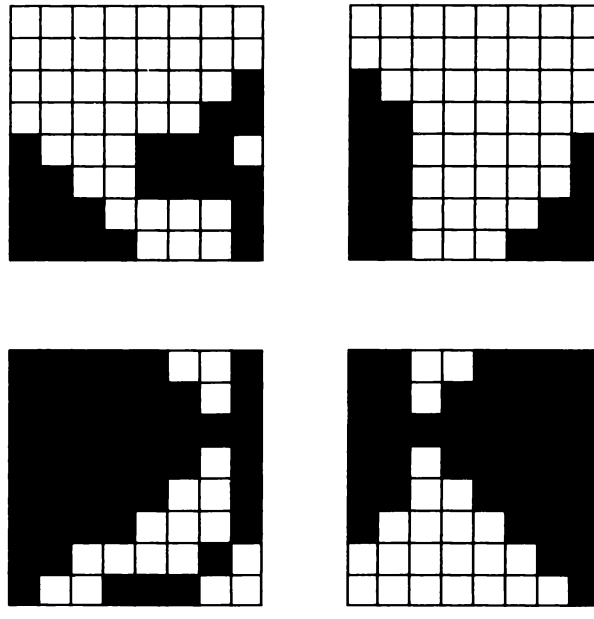


Figure 1.5 Werewolf

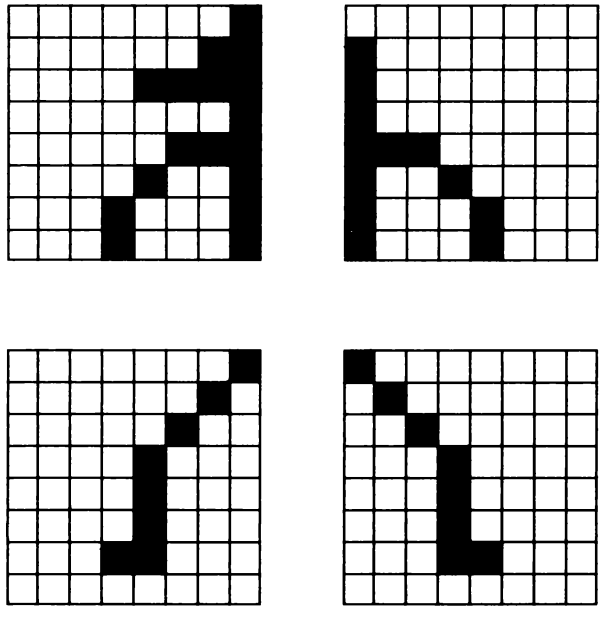


Figure 1.6 Wraith

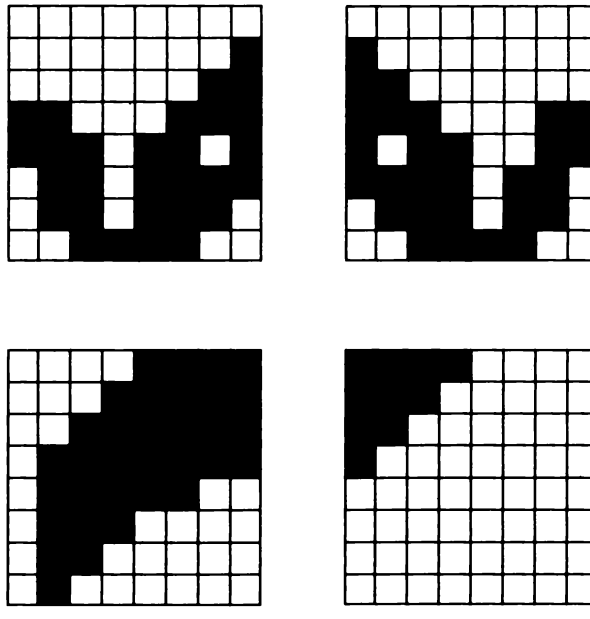
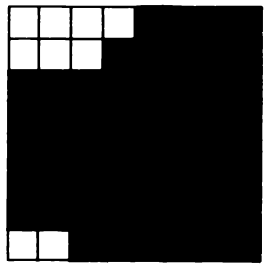
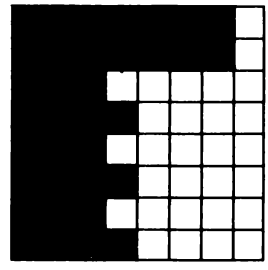


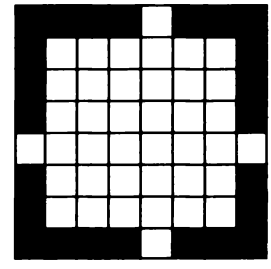
Figure 1.7



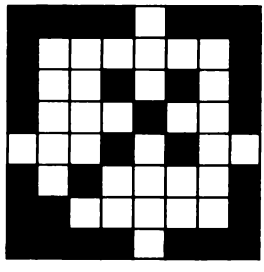
WIZARD'S HAND (Left)



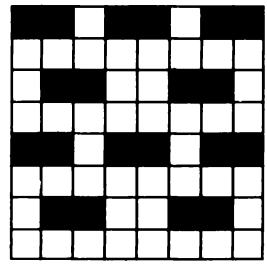
WIZARD'S HAND (Right)



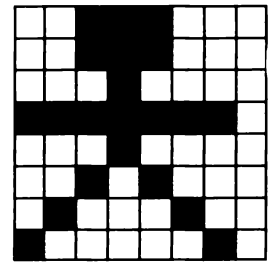
MAP (empty cave)



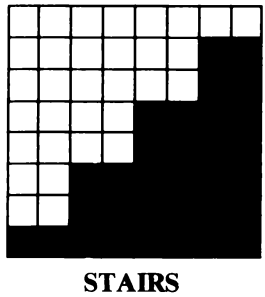
MAP (Treasure)



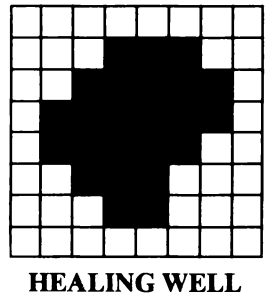
WALL



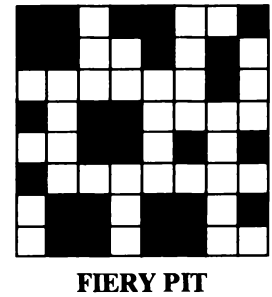
PLAYER



STAIRS

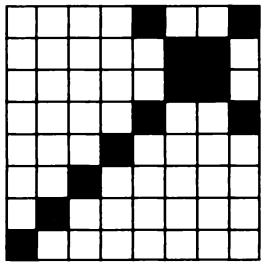


HEALING WELL

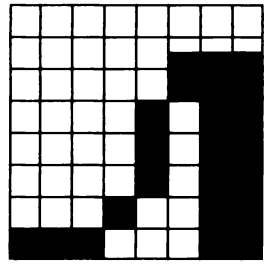


FIERY PIT

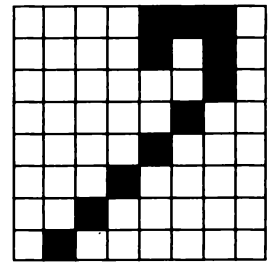
Figure 1.8



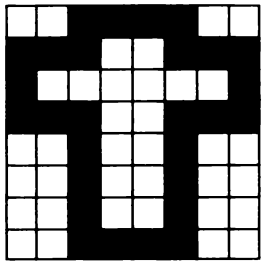
SWORD



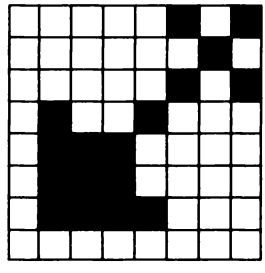
FIRE WHIP



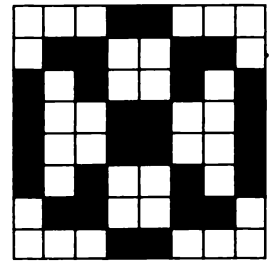
STAFF



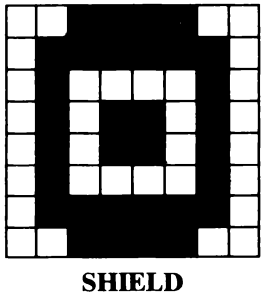
CROSS



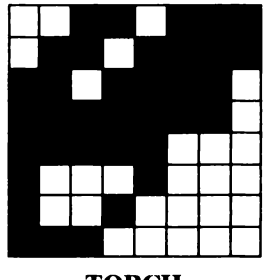
SPADE



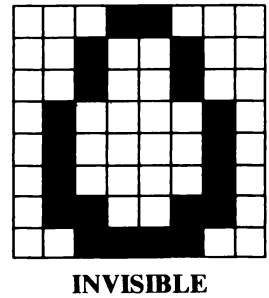
TALISMAN



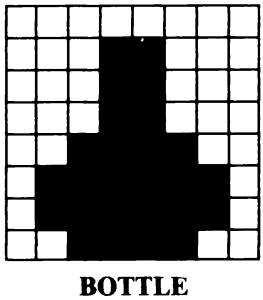
SHIELD



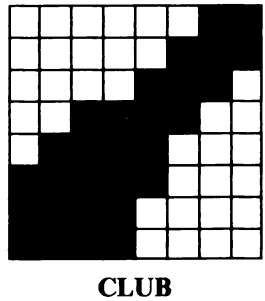
TORCH



INVISIBLE CLOAK



BOTTLE



CLUB

APPENDIX B
Instructions

The Eye of the Star Warrior

The aim of the game is to win possession of the enchanted stone (The Eye of the Star Warrior) and to destroy it.

You can move your player around the temple with the arrow keys 5,6,7&8. Movement is in the direction of the arrows.

To win you must manipulate your environment by using the following command words.

TAKE:	Allows you to pick up objects at your location.
DROP:	Allows you to drop objects you have in your possession. This command must be followed by name of object eg DROP SILVER SWORD.
I or INVENT:	Lists out objects and spells in your possession.
LOOK:	Lists out details of room.
DIG:	Allows you to dig for buried treasure.
MAP:	Prints out map showing the floor level you are exploring.
	Cave with treasure — Yellow
	Stored treasure — Green
	No treasures —
	Monster in cave — Cave flashes
	Player — P
WAIT:	Halts the game until you press Enter.
DRINK:	Allows you to drink any healing water that you might carry. If you don't carry healing water, the command will allow you to drink holy water.
UP:	Allows you to climb to an upper floor.
DOWN:	Allows you to descend to a lower floor.
TELEPORT:	When you have found the Teleport spell this command will allow you to escape danger by randomly placing you someplace else in the temple.

The table; printed top right on your sreen display shows your rating (Jester, Hero etc), strength, floor, status mode, combat rating and wounds.

Strength:	Amount of power you have. This value increases by killing monsters. You lose strength when wandering through the temple or during combat. This value must be kept above zero or you die of exhaustion.
Floor:	Shows you which of the three floors of the temple you are on.
STATUS MODE:	Green when there is no monster about. Yellow when there is a monster in the room. Red during combat.
Combat Rating:	Shows your percentage chance of killing the monster if you choose the proper weapon. eg if your combat rating reads 20 then you have a 20% chance of destroying the monster. The more damage you inflict on the monster the higher the value will become. The more damage the monster inflicts on you the lower the value will become. The lower the combat rating value is when you overcome the monster the more strength points you will gain.
WOUNDS:	Shows how many times you have been wounded. You are allowed a maximum of three wounds. The fourth wound is fatal. If you are wounded your strength will drain faster than normal, your foe will be more difficult to defeat and your weapons will be more prone to damage. You can set the wounds value back to zero by passing over a healing well or by drinking healing water from the well if you carry some with you.

Combat

If your player lands on a location occupied by a monster you will have first strike at the monster. If the monster moves onto your location then you will be asked to defend yourself. Not only will the computer accept the weapon you wish to use but it will also accept the following commands during combat: "INVENT" and "LOOK".

You can break out of the combat sequence by entering "N" to move 1 step north, "S" to move 1 step south, "W" to move 1 step west and "E" to move 1 step east. Once you are out of the combat sequence normal movement is achieved with the arrow keys. You can only break out of the combat sequence when it is your turn to attack unless you have and use a "TELEPORT" spell.

Combat with the Lich

Until he grows weak enough to allow you to attack him the Lich will throw spells at you and your only chance is to defend yourself or run by using the teleport spell. When the Lich grows weak enough you will be given your chance to attack.

After defeating the Lich, the Wizard's Hand that you have been carrying, will disappear, to be replaced by The Eye of the Star Warrior. Your task now is to destroy it by hurling it into the Fiery Pit, which is situated on the second level of the Dungeon complex.

SPELLS:	You will find spells buried in the temple. Using a spell will drain you of 50 strength units.
Wizard's Hand:	When you find the Wizard's Hand you will have a ghostly friend who will point to the location of buried treasures or spells.
Difficulty Level:	The higher you climb in the temple the faster the monsters will become.
Mind Vampire:	Beware of Mind Vampires. They feed on your memory and you will lose the spells you have been taught.
SAVE:	To save a game type in SAVE and press enter. Start your recorder and press any key. The computer will now save your game. When it is finished keep your tape running and press any key again — this time to save the graphics. When this is finished keep your tape running and press any key again — this time to save the screen display. When this is finished your game will resume where you left off. To load the saved game back into the computer, use LOAD"" or LOAD "adventure".

APPENDIX C
The Eye of the Star Warrior Listing

Throughout the listing, we have used REM statements to highlight the various modules. The program *will not run* with these left in, so crammed is the 48K! They are here purely for guidance, and *must* be ignored when typing in the program.

My printer will not reproduce the defined graphics used during the game, so we will adopt the convention of lower-case, underlined characters. For example, the character that will, when the program is RUN, become the staircase graphic, is to be found on the "k" key, and should be entered into the listing by pressing CAPS SHIFT together with the 9 key (thus putting the machine into graphic mode), and then the "k" key. Similarly, the "m" key will yield the UDG for the Fire Pit, and so on. When the program is eventually RUN, the graphic character will replace the letter in the listing.

```
  10 BORDER 0: INK 0: PAPER 0: C
LS : PRINT  INK 2; BRIGHT 1; FLA
SH 1;AT 10,0;"
                Adventure Loadin9:
 Do Not StoP  TaPe.

                "
  15 REM ** SET GRAPHICS **
  20 FOR a=0 TO 7
  30 REM .. WIZARDS HAND
  35 READ b: POKE USR "a"+a,b
  40 READ b: POKE USR "b"+a,b
  45 REM ..MAP ROOM WITHOUT
           TREASURE
  50 READ b: POKE USR "c"+a,b
  55 REM ..MAP ROOM WITH
           TREASURE
  60 READ b: POKE USR "d"+a,b
  70 REM .. STONE WALL
  80 READ b: POKE USR "i"+a,b
  85 REM .. PLAYER
  90 READ b: POKE USR "j"+a,b
 100 REM *STAIRS*
 110 READ b: POKE USR "k"+a,b
 120 REM *HEALING WELL*
 130 READ b: POKE USR "l"+a,b
 140 REM *FIRE PIT*
 150 READ b: POKE USR "m"+a,b
 160 NEXT a
 170 DATA BIN 00001111,BIN 11111
110,BIN 11110111,BIN 11110111,BI
N 11011011,BIN 00111000,BIN 0,BI
N 0,BIN 11011001
 180 DATA BIN 00011111,BIN 11111
110,BIN 10000001,BIN 10000001,BI
N 0,BIN 00111000,BIN 00000011,BI
N 00011100,BIN 11001010
 190 DATA BIN 11111111,BIN 11100
000,BIN 10000001,BIN 10010101,BI
N 01100110,BIN 00010000,BIN 0000
0011,BIN 00111110,BIN 00000010
 200 DATA BIN 11111111,BIN 11110
000,BIN 10000001,BIN 10001001,BI
N 0,BIN 11111110,BIN 00001111,BI
```

```
N 01111110,BIN 10110000
 210 DATA BIN 11111111,BIN 11100
000,BIN 0,BIN 00010100,BIN 11011
011,BIN 00010000,BIN 00001111,BI
N 01111100,BIN 00110101
 220 DATA BIN 11111111,BIN 11110
000,BIN 10000001,BIN 10100001,BI
N 0,BIN 00101000,BIN 00111111,BI
N 00111000,BIN 10000000
 230 DATA BIN 11111111,BIN 11100
000,BIN 10000001,BIN 11000001,BI
N 01100110,BIN 01000100,BIN 0011
1111,BIN 00011000,BIN 01101101
 240 DATA BIN 00111111,BIN 11110
000,BIN 11110111,BIN 11110111,BI
N 0,BIN 10000010,BIN 11111111,BI
N 0,BIN 01101100
 260 LOAD "prog"
 270 SAVE "adventure" LINE 10


   1 REM *ADVENTURE
         **@ Roy  Carnell
       -- Tony Bridge 1983**
   5 BORDER 0: PAPER 0: INK 6: C
LS : PRINT AT 11,0;" Press ENTER
 to set up dungeon. "
  10 PAUSE 0
  15 RANDOMIZE
  20 PRINT INK 2; BRIGHT 1; FLAS
H 1;AT 10,0;"
              PLEASE WAIT.SETTIN
G UP DUNGEON
            "
  30 POKE 23658,8
  40 GO TO 1640
  49 REM ************************
  50 REM CLEAR BOTTOM 8 LINES
         OF SCREEN
  51 REM ************************
  60 GO SUB 530
  70 FOR d=14 TO 21
  80 PRINT AT d,0;"
                   "
```

```
  90 NEXT d
 100 PRINT AT 13,0;"_"
 110 RETURN
 119 REM ************************
 120 REM Subroutine to restore
       data for weaPon graPhics
 121 REM ************************
 130 IF a=1 THEN RESTORE 2960
 140 IF a=2 THEN RESTORE 2980
 150 IF a=3 OR a=4 THEN RESTORE
3000
 160 IF a=5 OR a=6 THEN RESTORE
3020
 170 IF a=7 THEN RESTORE 3040
 180 IF a=8 THEN RESTORE 3060
 190 IF a=9 THEN RESTORE 3080
 200 IF a=10 THEN RESTORE 3100
 210 IF a=11 THEN RESTORE 3120
 220 IF a=12 THEN RESTORE 3140
 230 IF a=13 OR a=16 OR a=17 THE
N RESTORE 3160
 240 IF a=14 THEN RESTORE 3180
 250 IF a=15 THEN RESTORE 3200
 260 RETURN
 269 REM ************************
 270 REM Subroutine to restore
   data for monster toP graPhics
 271 REM ************************
 280 IF a=1 THEN RESTORE 3320
 290 IF a=2 OR a=4 OR a=6 OR a=1
6 THEN RESTORE 3360
 300 IF a=3 THEN RESTORE 3440
 310 IF a=5 THEN RESTORE 3260
 320 IF a=7 THEN RESTORE 3400
 330 IF a=8 THEN RESTORE 3460
 340 IF a=9 THEN RESTORE 3300
 350 IF a=10 THEN RESTORE 3240
 360 IF a=11 THEN RESTORE 3340
 370 IF a=12 THEN RESTORE 3220
 380 IF a=13 THEN RESTORE 3420
 390 IF a=14 THEN RESTORE 3280
 400 IF a=15 THEN RESTORE 3380
 410 RETURN
```

```
 419 REM **********************
 420 REM Subroutine to restore
data for monster bottom graphics
 421 REM **********************
 430 IF a=2 OR a=3 OR a=4 OR a=6
 OR a=7 OR a=11 OR a=15 OR a=16
THEN RESTORE 3600
 440 IF a=1 THEN RESTORE 3560
 450 IF a=5 THEN RESTORE 3540
 460 IF a=8 THEN RESTORE 3640
 470 IF a=9 THEN RESTORE 3620
 480 IF a=10 THEN RESTORE 3480
 490 IF a=12 THEN RESTORE 3500
 500 IF a=13 THEN RESTORE 3580
 510 IF a=14 THEN RESTORE 3520
 520 RETURN
 524 REM **********************
 525 REM PRINT STATUS TABLE
 526 REM **********************
 530 IF STRENGTH<=100 THEN PRINT
 PAPER 6; INK 0; BRIGHT 1;AT 2,1
6;"    MISFIT    "
 540 IF STRENGTH>100 AND STRENGT
H<=250 THEN PRINT PAPER 5; INK 0
; BRIGHT 1;AT 2,16;"    JESTER
   "
 550 IF STRENGTH>250 AND STRENGT
H<=400 THEN PRINT PAPER 4; INK 0
; BRIGHT 1;AT 2,16;"     HERO
   "
 560 IF STRENGTH>400 AND STRENGT
H<=500 THEN PRINT PAPER 3; INK 7
; BRIGHT 1;AT 2,16;"   WARRIOR
   "
 570 IF STRENGTH>500 AND SPELL=0
 THEN PRINT PAPER 7; INK 1; BRIG
HT 1;AT 2,16;"  SUPER HERO  "
 580 IF STRENGTH<=500 THEN GO TO
 610
 590 IF SPELL>0 AND SPELL<=3 THE
N PRINT PAPER 6; INK 0; BRIGHT 1
;AT 2,16;"    WIZARD    "
 600 IF SPELL>3 THEN PRINT PAPER
 2; INK 1; BRIGHT 1; FLASH 1;AT
```

```
2;16;" GRAND WIZARD "
 610 PRINT PAPER 7;AT 4,28;"
"
 620 IF STRENGTH>9999 THEN LET S
TRENGTH=9999
 630 PRINT PAPER 7; INK 0;AT 4,2
8;STRENGTH
 640 PRINT PAPER 7; INK 0;AT 6,2
9;FLOOR
 650 IF M(LOCATION)=0 THEN PRINT
 PAPER 4; INK 0; BRIGHT 1;AT 8,2
6;" GREEN"
 660 IF COMBAT>0 THEN GO TO 680
 670 IF M(LOCATION)<>0 THEN PRIN
; PAPER 6; INK 1; BRIGHT 1; FLAS
H 1;AT 8,26;"YELLOW"
 680 IF COMBAT>0 THEN PRINT INK
2; PAPER 7; BRIGHT 1; FLASH 1;AT
 8,26;"COMBAT"
 690 PRINT PAPER 7; INK 0;AT 10,
28;"     "
 700 IF PER>0 THEN PRINT PAPER 7
; INK 0;AT 10,29;PER
 710 PRINT PAPER 7; INK 0;AT 12,
29;WOUNDS
 720 IF STRENGTH<=0 THEN GO TO 7
50
 730 PRINT AT 13,0;"_"
 740 RETURN
 750 GO SUB 70
 760 PRINT "You have died of exh
austion."
 770 GO TO 6850
 779 REM **********************
 780 REM PRINT OUT CONTENTS OF
         ROOM
 781 REM **********************
 790 LET r=T(LOCATION)
 800 IF r=0 OR r=99 THEN GO TO 8
30
 810 IF r=18 THEN GO TO 980
 820 GO TO 840
 830 IF S(LOCATION)=0 THEN GO TO
 920
```

```
 840 PRINT "You have found: ";
 850 IF r=0 OR r=99 THEN GO TO 8
70
 860 PRINT T$(r)
 870 LET r=S(LOCATION)
 880 IF r=0 THEN GO TO 920
 890 IF T(LOCATION)=0 OR T(LOCAT
ION)=99 THEN GO TO 910
 900 PRINT "and ";
 910 PRINT T$(r)
 920 IF COMBAT=3 THEN GO TO 940
 925 IF M(LOCATION)<>0 THEN PRIN
T "The room is guarded by a"
 930 IF M(LOCATION)<>0 THEN PRIN
T M$(M(LOCATION))
 940 IF X$(3,10)="k" THEN PRINT
"You see stairs in the corner."
 950 IF X$(3,4)="l" THEN PRINT "
You see a healing well."
 960 IF X$(3,4)="m" THEN PRINT "
You see a fire pit."
 970 RETURN
 980 PRINT "You have found the g
ostly hand of the good Wizard.
It is yours until you find the e
ye."
 990 FOR r=1 TO 12
1000 IF H(r)=0 THEN GO TO 1020
1010 NEXT r
1020 LET H(r)=18
1030 LET WHAND=1
1040 LET T(LOCATION)=99
1050 GO TO 920
1059 REM **********************
1060 REM CHANGE TREASURE INPUT
         INTO NUMBERS
1061 REM **********************
1070 DIM V$(LEN S$)
1080 FOR r=1 TO 25
1090 LET V$=T$(r)
1100 IF V$=S$ THEN RETURN
1110 NEXT r
1120 LET r=0
1130 RETURN
```

```
1139 REM **********************
1140 REM CHECK FOR TREASURE
1141 REM **********************
1150 LET TRE=0
1160 FOR b=1 TO 12
1170 IF H(b)=r THEN GO TO 1200
1180 NEXT b
1190 RETURN
1200 LET TRE=1
1210 RETURN
1219 REM **********************
1220 REM CHECK YOUR WEAPONS
         EFFECT
1221 REM **********************
1230 LET B$=A$(M(LOCATION))
1240 FOR a=1 TO 20 STEP 2
1250 LET S$=B$(a TO a+1)
1260 IF S$="  " THEN GO TO 1290
1270 IF VAL S$=r THEN RETURN
1280 NEXT a
1290 LET r=0
1300 RETURN
1309 REM **********************
1310 REM CHECK YOUR DEFENCE
1311 REM **********************
1320 LET B$=D$(M(LOCATION))
1330 FOR a=1 TO 18 STEP 2
1340 LET S$=B$(a TO a+1)
1350 IF S$="  " THEN GO TO 1380
1360 IF VAL S$=r THEN RETURN
1370 NEXT a
1380 LET r=0
1390 RETURN
1399 REM **********************
1400 REM MIND VAMPIRE
1401 REM **********************
1410 GO SUB 60
1420 FOR r=1 TO 12
1430 IF H(r)>18 THEN LET H(r)=0
1440 NEXT r
1445 LET SPELL=0
1450 PRINT "Your mind is bein9 d
rained."
1460 RETURN
```

```
1470 PRINT INK 2; BRIGHT 1; FLAS
H 1;"        I dont understand.
     "
1480 FOR a=1 TO 25: BEEP 0.1,10:
 BEEP 0.1,30: BEEP 0.1,20: NEXT
a
1490 GO SUB 60
1500 RETURN
1509 REM **********************
1510 REM DAMAGE WEAPONS
1511 REM **********************
1520 IF WEAPON=13 OR WEAPON=17 T
HEN GO TO 1570
1525 IF WEAPON>16 THEN RETURN
1530 LET r=INT (RND*90)+1
1540 IF r+(WOUNDS*10)+(COMBAT*10
)<90 THEN RETURN
1550 PRINT "You have damaged the
"
1560 PRINT T$(WEAPON)
1565 PAUSE 100
1570 FOR a=1 TO 12
1580 IF H(a)=WEAPON THEN GO TO 1
600
1590 NEXT a
1600 IF WEAPON=13 OR WEAPON=17 T
HEN GO TO 1625
1605 LET H(a)=0
1610 LET HELD=HELD-1
1620 RETURN
1625 LET H(a)=16
1630 RETURN
1634 REM **********************
1635 REM SET UP DUNGEON
1636 REM **********************
1640 DIM C(300)
1650 FOR a=1 TO 10: LET C(a)=999
: NEXT a
1660 FOR a=90 TO 110: LET C(a)=9
99: NEXT a
1670 FOR a=190 TO 210: LET C(a)=
999: NEXT a
1680 FOR a=290 TO 300: LET C(a)=
999: NEXT a
```

```
1690 FOR a=11 TO 291 STEP 10: LE
T C(a)=999: NEXT a
1700 FOR a=10 TO 300 STEP 10: LE
T C(a)=999: NEXT a
1710 LET b=INT (RND*30)+30
1720 FOR a=1 TO b
1730 LET d=INT (RND*280)+10
1740 IF C(d)=999 THEN GO TO 1730
1750 IF C(d-10)=999 OR C(d+1)=99
9 OR C(d+11)=999 THEN GO TO 1730
1760 IF C(d-11)=999 OR C(d-9)=99
9 THEN GO TO 1780
1770 LET C(d)=999
1780 NEXT a
1790 LET d=0
1800 FOR a=10 TO 290
1810 IF C(a)=999 THEN GO TO 1840
1820 LET d=d+1
1830 LET C(a)=d
1840 NEXT a
1849 REM **********************
1850 REM Populate dungeon with
       monsters
1851 REM **********************
1860 DIM M(290)
1870 FOR a=10 TO 290
1880 LET b=INT (RND*4)+1
1890 IF b<2 THEN GO TO 1910
1900 LET M(a)=INT (RND*15)+1
1910 NEXT a
1920 LET M(12)=0
1930 LET a=INT (RND*80)+210
1940 IF C(a)=999 THEN GO TO 1930
1950 LET M(a)=16
1959 REM **********************
1960 REM Place treasures in
       dungeon
1961 REM **********************
1970 DIM T(290)
1980 FOR a=10 TO 290
1990 LET b=INT (RND*4)+1
2000 IF b<3 THEN GO TO 2020
2010 LET T(a)=INT (RND*15)+1
```

```
2020 NEXT a
2030 REM * ADD WIZARDS HAND *
2040 LET a=INT (RND*70)+20
2050 IF C(a)=999 THEN GO TO 2040
2060 LET T(a)=18
2067 REM **********************
2068 REM STORAGE SPACE
2069 REM **********************
2070 DIM S(290)
2079 REM **********************
2080 REM READ IN MONSTERS
2081 REM **********************
2090 DIM M$(19,15)
2100 FOR a=1 TO 19: READ M$(a):
NEXT a
2110 DATA "Living Skeleton","Mum
my","Demon","Zombie","Fire Eleme
ntal","VamPire","Mind VamPire","
Wraith","Dragon","Werewolf","Cyc
lops","Sandman","Harpie","Giant
Serpent","Balrog","Lich","Lightn
ing bolt","Stone sPell","Limbo s
pell"
2119 REM **********************
2120 REM READ IN TREASURES
2121 REM **********************
2130 DIM T$(25,15)
2140 FOR a=1 TO 25: READ T$(a):
NEXT a
2150 DATA "SPADE","FIRE WHIP","S
WORD","SILVER SWORD","SILVER STA
FF","SAINTLY STAFF","TALISMAN","
CROSS","SHIELD","TORCH","INVISIB
LE CLOAK","CLUB","HOLY WATER","B
OW AND ARROWS","MAGIC SHIELD","E
MPTY BOTTLE","HEALING WATER","WI
ZARDS HAND","TELEPORT","FORCESHI
ELD","PSYCHIC SHIELD","LIGHTNING
 BOLT","STONE SPELL","LIMBO SPEL
L","STONE"
2159 REM **********************
2160 REM READ IN ATTACK WEAPONS
          FOR EACH MONSTER
```

```
2161 REM **********************
2170 DIM A$(16,20)
2180 FOR a=1 TO 16: READ A$(a):
NEXT a
2190 DATA "0103040506071222 24","
020710222324","06071324","020710
2324","06071324","02040506101322
2324","0204050610132223 24","0607
24","0304222324","02040607102223
24","030414222324","010506071213
222324","02030414222324","020304
10222324","0304071422 2324","2324
"
2199 REM **********************
2200 REM READ IN DEFENCE
         WEAPONS AGAINST EACH
         MONSTER
2201 REM **********************
2210 DIM D$(19,18)
2220 FOR a=1 TO 19: READ D$(a):
NEXT a
2230 DATA "09111519","0209101115
19","0708151920","02070910111519
 ,"071319","020708091011131519",
"020708091011131519","0708151920
","09111519","0207091011131519",
"020910111519","091113151920","0
20910111519","020910111519","070
91519","192021","1520","21","21"
2239 REM **********************
2240 REM COMBAT RATINGS
2241 REM **********************
2250 DIM V(16)
2260 FOR a=1 TO 16: READ V(a): N
EXT a
2270 DATA 70,70,20,40,70,50,50,5
0,20,50,20,50,60,20,30,20
2279 REM **********************
2280 REM MONSTER COLOURS
2281 REM **********************
2290 DIM P(16)
2300 FOR a=1 TO 16: READ P(a): N
EXT a
```

```
2310 DATA 7,7,2,3,2,5,5,7,4,2,6,
6,4,4,3,1
2319 REM **********************
2320 REM TREASURE COLOURS
2321 REM **********************
2330 DIM W(17)
2340 FOR a=1 TO 17: READ W(a): N
EXT a
2350 DATA 7,2,7,5,5,6,3,6,6,2,3,
6,5,6,3,4,7
2359 REM **********************
2360 REM SET UP STAIRS, HEALING
       WELL and FIRE PIT
2361 REM **********************
2370 LET SONE=INT (RND*90)+10
2380 IF C(SONE)=999 THEN GO TO 2
370
2390 LET STWO=INT (RND*90)+110
2400 IF C(STWO)=999 THEN GO TO 2
390
2410 LET STHREE=INT (RND*90)+210
2420 IF C(STHREE)=999 THEN GO TO
 2410
2430 LET HW=INT (RND*90)+1
2440 IF C(HW)=999 THEN GO TO 243
0
2450 LET FP=INT (RND*90)+100
2460 IF C(FP)=999 THEN GO TO 245
0
2470 LET FLOOR=1: LET LOCATION=1
2: LET PX=6: LET PY=6: LET STREN
GTH=100: LET COMBAT=0: LET SPELL
=0: LET PER=0: LET WOUNDS=0: LET
 SM=0: LET HELD=0: LET RET=0: LE
T WHAND=0: LET STONE=0
2475 DIM H(12)
2480 DIM X$(12,12)
2490 FOR a=0 TO 12: PRINT PAPER
0; INK 1;AT a,0;"
                 ": NEXT a
2500 PRINT "____________________
____________"
2510 PRINT PAPER 1; INK 5;AT 0,1
9;"ADVENTURE"
```

```
2520 PRINT PAPER 1; INK 7;AT 4,1
9;"STRENGTH      "
2530 PRINT PAPER 1; INK 7;AT 6,2
2;"FLOOR      "
2540 PRINT PAPER 1; INK 7;AT 8,1
4;"STATUS MODE        "
2550 PRINT PAPER 1; INK 7;AT 10,
14;"COMBAT RATING      "
2560 PRINT PAPER 1; INK 7;AT 12,
21;"WOUNDS      "
2569 REM ********************
2570 REM SET UP AND PRINT
        GRAPHICS OF ROOM
2571 REM ********************
2580 LET X$(1,1 TO 12)="
     "
2590 LET X$(2,1 TO 12)=" iiiiiii
iii "
2600 FOR a=3 TO 10
2610 LET X$(a,1 TO 12)=" i
   i "
2620 NEXT a
2630 LET X$(11,1 TO 12)=" iiiiii
iiii "
2640 LET X$(12,1 TO 12)="
     "
2650 IF C(LOCATION-10)<>999 THEN
 LET X$(2,6 TO 7)="  "
2660 IF C(LOCATION+10)<>999 THEN
 LET X$(11,6 TO 7)="  "
2670 IF C(LOCATION-1)<>999 THEN
LET X$(6,2)=" "
2680 IF C(LOCATION-1)<>999 THEN
LET X$(7,2)=" "
2690 IF C(LOCATION+1)<>999 THEN
LET X$(6,11)=" "
2700 IF C(LOCATION+1)<>999 THEN
LET X$(7,11)=" "
2710 IF SONE=LOCATION OR STWO=LO
CATION OR STHREE=LOCATION THEN L
ET X$(3,10)="k"
2720 IF HW=LOCATION THEN LET X$(
3,4)="l"
```

```
2730 IF FP=LOCATION THEN LET X$(
3,4)="m"
2740 LET a=T(LOCATION)
2750 IF a=0 OR a>=18 THEN GO TO
2790
2760 GO SUB 130
2770 FOR a=0 TO 7: READ b: POKE
USR "o"+a,b: NEXT a
2780 IF RET=1 THEN RETURN
2790 LET a=S(LOCATION)
2800 IF a=0 THEN GO TO 2830
2810 GO SUB 130
2820 FOR a=0 TO 7: READ b: POKE
USR "p"+a,b: NEXT a
2830 IF RET=1 THEN RETURN
2840 LET a=M(LOCATION)
2850 IF a=0 THEN GO TO 3660
2860 GO SUB 280
2870 FOR a=0 TO 7
2880 READ b: POKE USR "q"+a,b
2890 READ b: POKE USR "u"+a,b
2900 NEXT a
2910 LET a=M(LOCATION): GO SUB 0
430
2920 FOR a=0 TO 7
2930 READ b: POKE USR "r"+a,b
2940 READ b: POKE USR "t"+a,b
2950 NEXT a
2960 REM **SPADE**
2970 DATA BIN 00000101,BIN 00000
010,BIN 00000101,BIN 01001000,BI
N 01110000,BIN 01110000,BIN 0111
1000,BIN 0
2980 REM **FIRE WHIP**
2990 DATA BIN 0,BIN 00000111,BIN
 10001011,BIN 10001011,BIN 10001
011,BIN 10010011,BIN 11100011,BI
N 0
3000 REM **SWORD**
3010 DATA BIN 00001001,BIN 00000
110,BIN 00000110,BIN 00001001,BI
N 00010000,BIN 00100000,BIN 0100
0000,BIN 10000000
3020 REM **STAFF**
```

```
3030 DATA BIN 00001110,BIN 00001
010,BIN 00000010,BIN 00000100,BI
N 00001000,BIN 00010000,BIN 0010
0000,BIN 01000000
3040 REM **TALISMAN**
3050 DATA BIN 00011000,BIN 01100
110,BIN 10100101,BIN 10011001,BI
N 10011001,BIN 10100101,BIN 0110
0110,BIN 00011000
3060 REM **CROSS**
3070 DATA BIN 00111100,BIN 11100
111,BIN 10000001,BIN 11100111,BI
N 00100100,BIN 00100100,BIN 0010
0100,BIN 00111100
3080 REM **SHIELD**
3090 DATA BIN 00011000,BIN 01111
110,BIN 11111111,BIN 11111111,BI
N 11111111,BIN 11111111,BIN 0111
1110,BIN 00011000
3100 REM **TORCH**
3110 DATA BIN 00110111,BIN 01101
111,BIN 11011110,BIN 11111110,BI
N 11111000,BIN 10001000,BIN 1001
0000,BIN 11100000
3120 REM **CLOAK**
3130 DATA BIN 00011000,BIN 00100
100,BIN 00100100,BIN 01000010,BI
N 01000010,BIN 01000010,BIN 0110
0110,BIN 00111100
3140 REM **CLUB**
3150 DATA BIN 00000011,BIN 00000
111,BIN 00001110,BIN 00111100,BI
N 01111000,BIN 11111000,BIN 1111
0000,BIN 11110000
3160 REM **HOLY WATER+BOTTLE**
3170 DATA BIN 0,BIN 00011000,BIN
 00011000,BIN 00011000,BIN 00111
100,BIN 01111110,BIN 01111110,BI
N 00111100
3180 REM **BOW & ARROWS**
3190 DATA BIN 01111100,BIN 01001
110,BIN 00100111,BIN 00010011,BI
N 01001001,BIN 00100101,BIN 0001
0011,BIN 00001000
```

```
3200 REM **ENCHANTED SHIELD**
3210 DATA BIN 00111100,BIN 01111
110,BIN 01000010,BIN 01011010,BI
N 01011010,BIN 01000010,BIN 0111
1110,BIN 00111100
3220 REM **SANDMAN TOP**
3230 DATA BIN 0,BIN 0,BIN 000000
11,BIN 11000000,BIN 00000111,BIN
 11100000,BIN 00001101,BIN 10110
000,BIN 00111111,BIN 11111100,BI
N 01111110,BIN 01111110,BIN 1100
1100,BIN 00110011,BIN 11001111,B
IN 11110011
3240 REM **WEREWOLF TOP**
3250 DATA BIN 00000001,BIN 0,BIN
 00000011,BIN 10000000,BIN 00001
111,BIN 10000000,BIN 00000001,BI
N 10000000,BIN 00000111,BIN 1110
0000,BIN 00001001,BIN 10010000,B
IN 00010001,BIN 10001000,BIN 000
10001,BIN 10001000
3260 REM **FIRE ELEMENTAL TOP**
3270 DATA BIN 00101101,BIN 00010
000,BIN 10100011,BIN 10100000,BI
N 10010111,BIN 11101100,BIN 0100
1111,BIN 11101001,BIN 01011101,B
IN 10110001,BIN 00111111,BIN 111
11010,BIN 00111110,BIN 01111100,
BIN 00111100,BIN 00111100
3280 REM **GIANT SERPENT TOP**
3290 DATA BIN 0,BIN 0,BIN 000000
01,BIN 10000000,BIN 00000001,BIN
 11100000,BIN 00000001,BIN 11110
000,BIN 00000011,BIN 10000000,BI
N 00000111,BIN 0,BIN 00001110,BI
N 0,BIN 00011111,BIN 11111000
3300 REM **DRAGON TOP**
3310 DATA BIN 0,BIN 0,BIN 0,BIN
0,BIN 00000001,BIN 10000000,BIN
00000011,BIN 11000000,BIN 100011
10,BIN 11000001,BIN 11001111,BIN
 11000001,BIN 11100001,BIN 11000
011,BIN 11110001,BIN 11000111
```

```
3320 REM **SKELETON TOP**
3330 DATA BIN 0,BIN 0,BIN 0,BIN
0,BIN 00000001,BIN 11000000,BIN
00000001,BIN 11000000,BIN 000000
01,BIN 11000000,BIN 0,BIN 100000
00,BIN 00000011,BIN 11100000,BIN
 00000100,BIN 10010000
3340 REM **CYCLOPS TOP**
3350 DATA BIN 0,BIN 0,BIN 0,BIN
0,BIN 00000001,BIN 10000000,BIN
00000011,BIN 11000000,BIN 000000
10,BIN 01000000,BIN 00000011,BIN
 11000000,BIN 00000001,BIN 10000
000,BIN 00000011,BIN 11000000
3360 REM **MUMMY,ZOMBIE,VAMPIRE,
LICH TOP**
3370 DATA BIN 0,BIN 0,BIN 0,BIN
0,BIN 00000001,BIN 10000000,BIN
00000011,BIN 11000000,BIN 000000
11,BIN 11000000,BIN 00000011,BIN
 11000000,BIN 00000001,BIN 10000
000,BIN 00000011,BIN 11000000
3380 REM **BALROG TOP**
3390 DATA BIN 00000010,BIN 01000
000,BIN 00000010,BIN 01000000,BI
N 00000010,BIN 01000000,BIN 0000
0011,BIN 11000000,BIN 00000011,B
IN 11000000,BIN 00000011,BIN 110
00000,BIN 00000001,BIN 10000000,
BIN 00000011,BIN 11000000
3400 REM **MIND VAMPIRE TOP**
3410 DATA BIN 00000001,BIN 10000
000,BIN 00000011,BIN 11000000,BI
N 00000111,BIN 11100000,BIN 0000
1101,BIN 10110000,BIN 00000111,B
IN 11100000,BIN 00000011,BIN 110
00000,BIN 00000001,BIN 10000000,
BIN 00000011,BIN 11000000
3420 REM **HARPIE TOP**
3430 DATA BIN 0,BIN 0,BIN 0,BIN
00000010,BIN 01000001,BIN 100001
10,BIN 01100011,BIN 11000111,BIN
 01110011,BIN 11001110,BIN 01110
```

```
011,BIN 11001111,BIN 01111001,BI
N 10011110,BIN 01111111,BIN 1111
1110
3440 REM **DEMON TOP**
3450 DATA BIN 00000010,BIN 01000
000,BIN 00000011,BIN 11000010,BI
N 01000010,BIN 01000110,BIN 0110
0011,BIN 11000111,BIN 01110011,B
IN 11001110,BIN 01110011,BIN 110
01111,BIN 01111001,BIN 10011110,
BIN 01111111,BIN 11111110
3460 REM WRAITH TOP**
3470 DATA BIN 0,BIN 0,BIN 000000
01,BIN 10000000,BIN 00000011,BIN
 11000000,BIN 11000111,BIN 11100
011,BIN 11101101,BIN 10110011,BI
N 01101111,BIN 11110110,BIN 0110
1110,BIN 01110110,BIN 00111100,B
IN 00111100
3480 REM **WEREWOLF BOTTOM**
3490 DATA BIN 00010001,BIN 10001
000,BIN 00000010,BIN 01000000,BI
N 00000100,BIN 00100000,BIN 0000
1000,BIN 00010000,BIN 00001000,B
IN 00010000,BIN 00001000,BIN 000
10000,BIN 00011000,BIN 00011000,
BIN 0,BIN 0
3500 REM **SANDMAN BOTTOM**
3510 DATA BIN 00001111,BIN 11110
000,BIN 00001100,BIN 00110000,BI
N 00011000,BIN 00011000,BIN 0011
0000,BIN 00001100,BIN 00110000,B
IN 00001100,BIN 00110000,BIN 000
01100,BIN 01110000,BIN 00001110,
BIN 0,BIN 0
3520 REM **GIANT SERPENT BOTTOM*
*
3530 DATA BIN 00011111,BIN 11111
000,BIN 00110000,BIN 0,BIN 00111
111,BIN 11111100,BIN 00111111,BI
N 11111100,BIN 01100000,BIN 0,BI
N 01111111,BIN 11111110,BIN 0111
1111,BIN 11111110,BIN 0,BIN 0
```

```
3540 REM **FIRE ELEMENTAL BOTTOM
**
3550 DATA BIN 10011110,BIN 01110
100,BIN 10111111,BIN 11111001,BI
N 01110000,BIN 11101101,BIN 0011
0010,BIN 11001110,BIN 01100100,B
IN 00001100,BIN 01111000,BIN 001
00110,BIN 01100000,BIN 00011110,
BIN 11100000,BIN 00000111
3560 REM **SKELETON BOTTOM**
3570 DATA BIN 00001000,BIN 10001
000,BIN 00010011,BIN 11100100,BI
N 00000001,BIN 01000000,BIN 0000
0001,BIN 01000000,BIN 00000001,B
IN 01000000,BIN 00000001,BIN 010
00000,BIN 00000001,BIN 01000000,
BIN 00000011,BIN 01100000
3580 REM **HARPIE BOTTOM**
3590 DATA BIN 01111011,BIN 11011
110,BIN 01110101,BIN 10101110,BI
N 01101101,BIN 10110110,BIN 0111
1101,BIN 10111110,BIN 01111010,B
IN 01011110,BIN 01110100,BIN 001
00110,BIN 01100100,BIN 00100010,
BIN 01001100,BIN 00110000
3600 REM **CYCLOPS*MUMMY*ZOMBIE*
VAMPIRE*BALROG*MIND VAMPIRE*LICH
BOTTOM**
3610 DATA BIN 00000101,BIN 10100
000,BIN 00001001,BIN 10010000,BI
N 00010001,BIN 10001000,BIN 0000
0001,BIN 10000000,BIN 00000010,B
IN 01000000,BIN 00000100,BIN 001
00000,BIN 00000100,BIN 00100000,
BIN 00001100,BIN 00110000
3620 REM *DRAGON BOTTOM*
3630 DATA BIN 11111001,BIN 11001
111,BIN 11111101,BIN 11011111,BI
N 11111111,BIN 11111111,BIN 1111
1101,BIN 11011111,BIN 11111001,B
IN 11001111,BIN 11110001,BIN 100
00111,BIN 11000010,BIN 00000011,
BIN 10011100,BIN 00000001
```

```
3640 REM *WRAITH BOTTOM*
3650 DATA BIN 00001111,BIN 11110
000,BIN 00011111,BIN 11100000,BI
N 00111111,BIN 11000000,BIN 0111
1111,BIN 10000000,BIN 01111100,B
IN 0,BIN 01110000,BIN 0,BIN 0110
0000,BIN 0,BIN 01000000,BIN 0
3660 LET WSC=0: LET WFC=0: LET M
C=0: LET SY=1: LET SX=1: LET WX=
1: LET WY=1: LET MY=0: LET MX=0:
 LET PER=0: LET BY=0: LET BX=0
3667 REM **********************
3668 REM PLACE OBJECTS IN NEW
        ROOM.
3669 REM **********************
3670 IF T(LOCATION)<1 OR T(LOCAT
ION)>17 THEN GO TO 3710
3680 LET WX=INT (RND*5)+4: LET W
Y=INT (RND*5)+4
3690 LET X$(WY,WX)="O"
3700 LET WFC=W(T(LOCATION))
3710 IF S(LOCATION)<1 THEN GO TO
 3760
3720 LET SX=INT (RND*5)+4: LET S
Y=INT (RND*5)+4
3730 IF SX=WX AND SY=WY THEN GO
TO 3720
3740 LET X$(SY,SX)="P"
3750 LET WSC=W(S(LOCATION))
3760 IF T(LOCATION)<>0 THEN GO T
O 3800
3770 IF T(LOCATION)=0 THEN LET B
Y=INT (RND*5)+4
3780 IF T(LOCATION)=0 THEN LET B
X=INT (RND*5)+4
3790 IF X$(BY,BX)<>" " THEN GO T
O 3760
3800 IF M(LOCATION)<1 THEN GO TO
 3820
3810 LET MC=P(M(LOCATION))
3820 PRINT AT 0,0;" "
3830 FOR a=1 TO 12
3840 PRINT TAB 1;X$(a,1 TO 12)
```

```
3850 NEXT a
3860 PRINT PAPER 0; INK 4; BRIGH
T 1;AT 12,3;"ROOM ";C(LOCATION);
"  "
3870 PRINT INK WFC; BRIGHT 1;AT
WY,WX;"O"
3880 PRINT INK WSC; BRIGHT 1;AT
SY,SX;"P"
3890 IF HW=LOCATION THEN PRINT I
NK 5;AT 3,4;"l"
3900 IF FP=LOCATION THEN PRINT I
NK 2; PAPER 6; BRIGHT 1; FLASH 1
;AT 3,4;"m"
3910 PRINT INK 6;AT PY,PX;"j"
3920 IF T(LOCATION)=0 AND WHAND=
1 THEN PRINT INK 4; BRIGHT 1;AT
BY,BX-2;"ab"
3930 IF M(LOCATION)<1 THEN GO TO
 3990
3940 LET MY=6: LET MX=6
3945 LET PER=V(M(LOCATION))
3950 LET PER=PER-(WOUNDS*5)
3960 IF PER<0 THEN LET PER=0
3970 IF PER>100 THEN LET PER=100
3980 PRINT INK MC; BRIGHT 1;AT M
Y,MX;"QU";AT MY+1,MX;"RT"
3990 GO SUB 60
4000 GO SUB 790
4010 GO SUB 530
4019 REM **********************
4020 REM INPUT YOUR COMMAND
4021 REM **********************
4030 LET a=0
4040 DIM B$(32)
4050 LET RET=0
4060 LET COMBAT=0
4070 LET k=0
4080 LET S$=INKEY$
4090 IF CODE S$=12 OR CODE S$=48

4090 IF CODE S$=12 OR CODE S$=48
4080>LET S$=INKEY$
4090 IF CODE S$=12 OR CODE S$=48
 THEN GO TO 4230
```

```
4100 IF CODE S$=13 THEN GO TO 42
00
4110 LET a=a+1
4120 IF a>=60/FLOOR THEN GO TO 4
930
4130 IF S$="" THEN GO TO 4080
4140 IF S$="5" OR S$="6" OR S$="
7" OR S$="8" THEN GO TO 4400
4150 IF k>=31 THEN GO TO 4080
4160 LET k=k+1
4170 LET B$(k)=S$: LET B$(k+1)="
>
4180 PRINT AT 21,0;B$: BEEP 0.1,
20
4190 LET a=a+1
4200 IF a>=60/FLOOR THEN GO TO 4
930
4210 IF INKEY$<>"" THEN GO TO 41
90
4220 GO TO 4080
4230 IF k<1 THEN GO TO 4080
4240 LET B$(k)=">": LET B$(k+1)=
 "
4250 LET k=k-1
4260 GO TO 4180
4270 GO SUB 60
4280 IF B$( TO 4)="TAKE" THEN GO
 TO 5120
4290 IF B$(1)="I" THEN GO TO 544
0
4300 IF B$( TO 4)="DROP" THEN GO
 TO 5590
4310 IF B$( TO 4)="WAIT" THEN GO
 TO 6070
4320 IF B$( TO 4)="LOOK" THEN GO
 TO 4850
4330 IF B$( TO 3)="DIG" THEN GO
TO 6250
4340 IF B$( TO 5)="DRINK" THEN G
O TO 6900
4350 IF B$( TO 2)="UP" OR B$( TO
 4)="DOWN" THEN GO TO 6140
```

```
4330 IF B$( TO 2)="UP" OR B$( TO
 4)="DOWN" THEN GO TO 6140
4355 IF B$( TO 3)="MAP" THEN GO
TO 8710
4360 IF B$( TO 8)="TELEPORT" THE
N GO TO 6720
4365 IF B$( TO 4)="SAVE" THEN GO
 TO 8500
4370 PRINT INK 5; BRIGHT 1;"I do
nt understand your command."
4380 GO TO 4040
4389 REM **********************
4390 REM MOVE PLAYER
4391 REM **********************
4400 LET TX=PX: LET TY=PY
4410 IF S$="W" THEN LET S$="5"
4420 IF S$="S" THEN LET S$="6"
4430 IF S$="N" THEN LET S$="7"
4440 IF S$="E" THEN LET S$="8"
4450 IF S$="5" THEN LET TX=TX-1
4460 IF S$="6" THEN LET TY=TY+1
4470 IF S$="7" THEN LET TY=TY-1
4480 IF S$="8" THEN LET TX=TX+1
4500 LET SM=SM+1
4510 IF SM<=8 THEN GO TO 4550
4520 LET STRENGTH=STRENGTH-(1+WO
UNDS)
4530 GO SUB 530
4540 LET SM=0
4550 IF X$(TY,TX)="i" THEN GO TO
 4040
4560 IF TY<2 OR TY>11 THEN GO TO
 4740
4570 IF TX<2 OR TX>11 THEN GO TO
 4740
4580 LET COL=6
4590 IF X$(PY,PX)="O" THEN LET C
OL=WFC
4600 IF X$(PY,PX)="P" THEN LET C
OL=WSC
4610 IF X$(PY,PX)="l" THEN LET C
OL=5
4620 PRINT INK COL;BRIGHT 1;AT P
Y,PX;X$(PY,PX)
```

```
4630 IF FP=LOCATION THEN  PRINT
 INK 2; PAPER 6; BRIGHT 1; FLASH
 1;AT 3,4;"m"
4640 IF T(LOCATION)=0 AND WHAND=
1 THEN  PRINT  INK 4; BRIGHT 1;A
T BY,BX-2;"ab"
4650 IF M(LOCATION)<>0 THEN  PRI
NT  INK P(M(LOCATION)); BRIGHT 1
;AT MY,MX;"qu";AT MY+1,MX;"rt"
4660 PRINT AT TY,TX;"j"
4665 LET PY=TY: LET PX=TX
4670 IF X$(PY,PX)<>"l" THEN  GO
TO 4635
4675 LET WOUNDS=0
4680 GO SUB 530
4685 IF STONE=1 AND X$(PY,PX)="m
" THEN  GO TO 7070
4690 IF PY=MY AND PX=MX OR PY=MY
 AND PX=MX+1 OR PY=MY+1 AND PX=M
X OR PY=MY+1 AND PX=MX+1 THEN  G
O TO 7120
4700 LET a=a+2+WOUNDS
4710 IF a>=60/FLOOR THEN  GO TO
4930
4720 GO TO 4040
4729 REM **********************
4730 REM MOVE TO NEW ROOM
4731 REM **********************
4740 IF S$="5" THEN  LET LOCATIO
N=LOCATION-1
4750 IF S$="5" THEN  LET PX=11
4760 IF S$="6" THEN  LET LOCATIO
N=LOCATION+10
4770 IF S$="6" THEN  LET PY=2
4780 IF S$="7" THEN  LET LOCATIO
N=LOCATION-10
4790 IF S$="7" THEN  LET PY=11
4800 IF S$="8" THEN  LET LOCATIO
N=LOCATION+1
4810 IF S$="8" THEN  LET PX=2
4820 FOR r=1 TO 11: PRINT AT r,0
;"                ": NEXT r
4830 GO TO 2580
4839 REM **********************
```

```
4840 REM LOOK COMMAND
4841 REM **********************
4850 GO SUB 60
4860 GO SUB 790
4870 IF COMBAT>0 THEN  GO TO 489
0
4880 GO TO 4040
4890 IF COMBAT=3 THEN  PRINT "Th
e Lich is attacking you with a";
M$(M(LOCATION))
4900 PAUSE 300
4905 IF COMBAT=1 THEN  GO TO 714
0
4910 IF COMBAT=2 THEN  GO TO 761
0
4915 GO TO 8090
4919 REM **********************
4920 REM MONSTER MOVEMENT
4921 REM **********************
4930 LET TY=MY: LET TX=MX
4935 LET a=0
4940 IF PY<TY THEN  LET TY=TY-1
4950 IF PY>TY THEN  LET TY=TY+1
4960 IF PX>TX THEN  LET TX=TX+1
4970 IF PX<TX THEN  LET TX=TX-1
4980 IF TY<3 OR TY>9 OR TX<3 OR
TX>9 THEN  GO TO 4030
4990 PRINT AT MY,MX;X$(MY,MX);AT
 MY,MX+1;X$(MY,MX+1)
5000 PRINT AT MY+1,MX;X$(MY+1,MX
);AT MY+1,MX+1;X$(MY+1,MX+1)
5010 PRINT  INK WFC;AT WY,WX;X$(
WY,WX)
5020 PRINT  INK WSC;AT SY,SX;X$(
SY,SX)
5030 IF HW=LOCATION THEN  PRINT
 INK 5;AT 3,4;"l"
5040 IF FP=LOCATION THEN  PRINT
 INK 2; PAPER 6; BRIGHT 1; FLASH
 1;AT 3,4;"m"
5050 IF T(LOCATION)=0 AND WHAND=
1 THEN  PRINT  INK 4; BRIGHT 1;A
T BY,BX-2;"ab"
5060 PRINT AT PY,PX;"j"
```

```
5070 PRINT  INK P(M(LOCATION));
BRIGHT 1;AT TY,TX;"qu";AT TY+1,T
X;"rt"
5080 LET MY=TY: LET MX=TX
5090 IF MY=PY AND MX=PX OR MY=PY
 AND MX+1=PX OR MY+1=PY AND MX=P
X OR MY+1=PY AND MX+1=PX THEN  G
O TO 7590
5100 GO TO 4080
5109 REM **********************
5110 REM TAKE OBJECT
5111 REM **********************
5120 IF X$(PY,PX)<>"l" THEN  GO
TO 5200
5130 FOR r=1 TO 12
5140 IF H(r)=16 THEN  GO TO 5170
5150 NEXT r
5160 PRINT "You need an empty bo
ttle to     collect water from t
he healing  well.": GO TO 4040
5170 LET H(r)=17
5180 PRINT "You have collected a
 bottle fullof healing water."
5190 GO TO 4040
5200 IF X$(PY,PX)="o" OR X$(PY,P
X)="p" THEN  GO TO 5230
5210 PRINT "I see nothing to Pic
k uP."
5220 GO TO 4040
5230 IF HELD<5 THEN  GO TO 5260
5240 PRINT "Sorry; you cannot ca
rry anymore treasures."
5250 GO TO 4040
5260 FOR r=1 TO 12
5270 IF H(r)=0 THEN  GO TO 5300
5280 NEXT r
5290 GO TO 5240
5300 IF X$(PY,PX)="o" THEN  LET
H(r)=T(LOCATION)
5310 IF X$(PY,PX)="p" THEN  LET
H(r)=S(LOCATION)
5320 PRINT "You have Picked uP:"
5330 IF X$(PY,PX)="o" THEN  PRIN
T I$(T(LOCATION))
```

```
5340 IF X$(PY,PX)="p" THEN  PRIN
T T$(S(LOCATION))
5350 LET HELD=HELD+1
5360 IF X$(PY,PX)="o" THEN  LET
T(LOCATION)=99
5370 IF X$(PY,PX)="p" THEN  LET
S(LOCATION)=0
5380 LET X$(PY,PX)=" "
5390 LET r=INT (RND*290)+1
5400 IF O(r)=999 OR T(r)=18 THEN
  GO TO 5390
5410 LET T(r)=INT (RND*15)+1
5420 GO TO 4040
5429 REM ******
5430 REM INVENT
5431 REM ******
5440 GO SUB 60
5450 LET d=13: LET r=0
5460 FOR b=1 TO 12
5470 IF H(b)=0 THEN  GO TO 5520
5480 LET d=d+1
5490 IF d=20 THEN  LET r=17
5500 IF d=20 THEN  LET d=14
5510 PRINT AT d,r;T$(H(b))
5520 NEXT b
5530 IF COMBAT>0 THEN  GO TO 555
0
5540 GO TO 4040
5550 PAUSE 500
5560 IF COMBAT=1 THEN  GO TO 714
0
5570 IF COMBAT=2 THEN  GO TO 761
0
5575 IF COMBAT=3 THEN  GO TO 809
0
5579 REM ****
5580 REM DROP
5581 REM ****
5590 LET RET=1
5600 IF PY=2 OR PY=11 OR PX=2 OR
 PX=11 THEN  GO TO 5620
5610 GO TO 5640
5620 PRINT "You cannot drop obje
cts in the  doorway."
```

```
5630 GO TO 4040
5640 LET B$(k+1)=""
5650 LET S$=B$(6 TO 20)
5660 FOR r=1 TO 22
5670 IF S$=T$(r) THEN  GO TO 571
0
5680 NEXT r
5690 PRINT "I don't understand w
hat you wantme to drop."
5700 GO TO 4040
5710 IF r<16 THEN  GO TO 5780
5720 IF r<>16 OR r<>25 THEN  GO
TO 5760
5730 PRINT "You cannot drop the"
5740 PRINT T$(r)
5750 GO TO 4040
5760 PRINT "You cannot drop some
thing that  exists in your memor
y."
5770 GO TO 4040
5780 FOR b=1 TO 12
5790 IF H(b)=r THEN  GO TO 5830
5800 NEXT b
5810 PRINT "You cannot drop some
thing you   do not carry."
5820 GO TO 4040
5830 LET H(b)=0
5840 IF S(LOCATION)=0 OR T(LOCAT
ION)=99 THEN  GO TO 5870
5850 PRINT "Invisible Imps carry
 off the    treasure you have ju
st dropped."
5860 GO TO 6040
5870 IF S(LOCATION)=0 THEN  GO T
O 5890
5880 GO TO 5960
5890 LET S(LOCATION)=r
5900 GO SUB 2790
5910 LET SX=INT (RND*5)+4: LET S
Y=INT (RND*5)+4
5920 IF X$(SY,SX)<>" " THEN  GO
TO 5910
5930 LET X$(SY,SX)="p": LET WSC=
W(r)
```

```
5940 PRINT  INK WSC; BRIGHT 1;AT
 SY,SX;X$(SY,SX)
5950 GO TO 6020
5960 LET T(LOCATION)=r
5970 GO SUB 2740
5980 LET WY=INT (RND*5)+4: LET W
X=INT (RND*5)+4
5990 IF X$(WY,WX)<>" " THEN  GO
TO 5980
6000 LET X$(WY,WX)="o": LET WFC=
W(r)
6010 PRINT  INK WFC; BRIGHT 1;AT
 WY,WX;X$(WY,WX)
6020 PRINT AT PY,PX;"j"
6030 IF M(LOCATION)<>0 THEN  PRI
NT  INK P(M(LOCATION)); BRIGHT 1
;AT MY,MX;"qu";AT MY+1,MX;"rt"
6040 LET HELD=HELD-1
6050 GO TO 4030
6059 REM ****
6060 REM WAIT
6061 REM ****
6070 GO SUB 60
6080 PRINT  INK 2; BRIGHT 1; FLA
SH 1;" WAITING "; INK 5; FLASH 0
;"Press Enter to continue"
6090 IF INKEY$<>"" THEN  GO TO 6
090
6100 IF INKEY$="" THEN  GO TO 61
00
6110 GO SUB 60
6120 GO TO 4040
6129 REM **********************
6130 REM CLIMB STAIRS
6131 REM **********************
6140 IF X$(PY,PX)="k" THEN  GO T
O 6170
6150 PRINT "You can't do that --
 you must  find the stairway."
6160 GO TO 4040
6170 IF B$( TO 2)="UP" AND LOCAT
ION=STHREE THEN  GO TO 6260
6180 IF B$( TO 4)="DOWN" AND LOC
ATION=SONE THEN  GO TO 6260
```

```
6190 IF B$( TO 2)="UP" AND LOCAT
ION=STWO THEN  LET LOCATION=STHR
EE
6200 IF B$( TO 2)="UP" AND LOCAT
ION=SONE THEN  LET LOCATION=STWO
6210 IF B$( TO 4)="DOWN" AND LOC
ATION=STWO THEN  LET LOCATION=SO
NE
6220 IF B$( TO 4)="DOWN" AND LOC
ATION=STHREE THEN  LET LOCATION=
STWO
6230 IF B$( TO 2)="UP" THEN  LET
 FLOOR=FLOOR+1
6240 IF B$( TO 4)="DOWN" THEN  L
ET FLOOR=FLOOR-1
6250 GO TO 2580
6260 PRINT "You can't go that wa
y."
6270 GO TO 4040
6279 REM ***
6280 REM DIG
6281 REM ***
6290 FOR r=1 TO 12
6300 IF H(r)=1 THEN  GO TO 6340
6310 NEXT r
6320 PRINT "You will need a spad
e."
6330 GO TO 4040
6340 IF BY=PY AND BX=PX THEN  GO
 TO 6370
6350 PRINT "There is nothing bur
ied here."
6360 GO TO 4040
6370 IF WHAND=1 THEN  GO TO 6400
6380 PRINT "You won't find burie
d treasure  without the Wizards
hand."
6390 GO TO 4040
6400 LET r=INT (RND*4)+1
6410 PRINT AT BY,BX-2;X$(BY,BX-2
);X$(BY,BX-1)
6420 GO SUB 60
6430 IF r>2 THEN  LET b=INT (RND
*6)+19
```

```
6440 IF r<=2 THEN  LET b=INT (RN
D*15)+1
6450 IF b>18 THEN  GO TO 6550
6460 LET T(LOCATION)=b
6470 PRINT "You have found: ";T$
(b)
6480 LET WY=BY: LET WX=BX
6490 LET BY=0: LET BX=0
6500 LET RET=1
6510 GO SUB 2740
6520 LET X$(WY,WX)="o"
6530 LET WFC=W(T(LOCATION))
6540 GO TO 4040
6550 PRINT "You can now use a:"
6560 PRINT T$(b)
6570 LET T(LOCATION)=99
6580 LET d=INT (RND*290)+1
6590 IF T(d)=0 OR C(d)=999 THEN
 GO TO 6580
6600 LET T(d)=0
6610 LET BY=0: LET BX=0
6620 FOR r=1 TO 12
6630 IF H(r)=b THEN  GO TO 4040
6640 NEXT r
6650 FOR r=1 TO 12
6660 IF H(r)=0 THEN  GO TO 6680
6670 NEXT r
6680 LET H(r)=b
6690 LET SPELL=SPELL+1
6700 GO TO 4040
6709 REM ********
6710 REM TELEPORT
6711 REM ********
6720 GO SUB 60
6730 FOR d=1 TO 12
6740 IF H(d)=19 THEN  GO TO 6780
6750 NEXT d
6760 PRINT "You don't know that
spell."
6770 GO TO 4040
6780 LET LOCATION=INT (RND*290)+
1
6790 IF C(LOCATION)=999 THEN  GO
 TO 6780
```

```
6800 LET PY=3: LET PX=3:
6810 LET STRENGTH=STRENGTH-50
6820 GO SUB 530
6830 GO TO 2560
6839 REM *******
6840 REM RESTART
6841 REM *******
6850 PRINT "Press ENTER to start
 new game."
6860 IF INKEY$<>"" THEN  GO TO 6
860
6870 IF INKEY$="" THEN  GO TO 68
70
6880 RUN
6889 REM *****
6890 REM DRINK
6891 REM *****
6900 FOR r=1 TO 12
6910 IF H(r)=17 THEN  GO TO 6980
6920 NEXT r
6930 FOR r=1 TO 12
6940 IF H(r)=13 THEN  GO TO 7030
6950 NEXT r
6960 PRINT "You can't drink when
 you carry  no water."
6970 GO TO 4040
6980 LET WOUNDS=0
6990 LET H(r)=16
7000 GO SUB 60
7010 PRINT "You drink a bottle o
f healing  water."
7020 GO TO 4040
7030 LET H(r)=16
7040 PRINT "You drink a bottle o
f holy water"
7050 GO TO 4040
7059 REM **********************
7060 REM END OF GAME
7061 REM **********************
7070 PRINT  INK 2; PAPER 0; BRIG
HT 1; FLASH 1;AT PY,PX;" "
7080 GO SUB 60
```

```
7090 PRINT "The eye is destroyed
 and the    land will remain fre
e of evil."
7100 GO TO 6850
7109 REM **********************
7110 REM COMBAT ATTACK ROUTINE
7111 REM **********************
7120 PRINT  INK P(M(LOCATION));
BRIGHT 1;AT MY,MX;"qu";AT MY+1,M
X;"rt"
7130 IF M(LOCATION)=16 THEN  GO
TO 8050
7140 LET COMBAT=1: GO SUB 50
7150 PRINT "Choose a weapon and
attack."
7160 IF INKEY$<>"" THEN  GO TO 7
160
7170 INPUT S$
7180 IF S$="W" OR S$="S" OR S$="
N" OR S$="E" THEN  GO TO 4400
7190 IF S$="I" OR S$="INVENT" TH
EN  GO TO 5440
7200 IF S$="LOOK" THEN  GO TO 48
50
7210 IF LEN S$>32 THEN  GO TO 72
30
7220 PRINT  INK 5; BRIGHT 1;S$
7230 GO SUB 1070
7240 IF r>0 THEN  GO TO 7270
7250 GO SUB 1470
7260 GO TO 7150
7270 GO SUB 1150
7280 IF TRE=1 THEN  GO TO 7310
7290 PRINT "You don't have that
weapon."
7300 GO TO 7610
7310 IF S$="TELEPORT" THEN  GO T
O 6720
7320 LET STRENGTH=STRENGTH-(WOUN
DS+1)
7330 IF r>18 THEN  LET STRENGTH=
STRENGTH-49
7340 LET WEAPON=r
7350 GO SUB 1230
```

```
7360 IF r>0 THEN  GO TO 7410
7370 PRINT "Your weaPon is usele
ss a9ainst"
7380 PRINT "the ";M$(M(LOCATION)
); "."
7390 GO SUB 1520
7400 GO TO 7610
7410 IF INT (RND*100)+1<PER THEN
  GO TO 7470
7420 PRINT "You have wounded the
"
7430 PRINT M$(M(LOCATION))
7440 LET PER=PER+10
7450 IF PER>100 THEN  LET PER=10
0
7455 GO SUB 1520
7460 GO TO 7610
7470 IF M(LOCATION)=16 THEN  GO
TO 8210
7475 PRINT "You have destroyed t
ne"
7480 PRINT M$(M(LOCATION))
7485 GO SUB 1520
7490 LET M(LOCATION)=0
7500 PRINT AT MY,MX;X$(MY,MX);X$
(MY,MX+1)
7510 PRINT AT MY+1,MX;X$(MY+1,MX
);X$(MY+1,MX+1)
7520 PRINT AT PY,PX;"j"
7530 PAUSE 100
7540 LET STRENGTH=STRENGTH+(100-
PER)
7550 LET COMBAT=0: LET PER=0
7560 GO SUB 0050
7570 LET MY=0: LET MX=0: LET P=0
7580 GO TO 4030
7589 REM **********************
7590 REM COMBAT DEFENCE
7591 REM **********************
7600 IF M(LOCATION)=16 THEN  GO
TO 6060
7610 GO SUB 60
7620 LET COMBAT=2
```

```
7630 IF M(LOCATION)=7 THEN  GO S
UB 1410
7640 PRINT "You are under attack
 choose yourdefence."
7650 IF INKEY$<>"" THEN  GO TO 7
650
7660 INPUT S$
7670 IF S$="I" OR S$="INVENT" TH
EN  GO TO 5440
7680 IF S$="LOOK" THEN  GO TO 48
50
7690 GO SUB 60
7700 PRINT "Choose your defence.
"
7710 IF LEN S$>32 THEN  GO TO 77
30
7720 PRINT  INK 5; BRIGHT 1;S$
7730 GO SUB 1070
7740 IF r>0 THEN  GO TO 7770
7750 GO SUB 1470
7760 GO TO 7640
7770 GO SUB 1150
7780 IF TRE=1 THEN  GO TO 7820
7790 PRINT "You cant use that."
7800 IF COMBAT=3 THEN  GO TO 803
0
7810 GO TO 7980
7820 IF S$="TELEPORT" THEN  GO T
O 6720
7830 LET WEAPON=r
7840 LET STRENGTH=STRENGTH-(1+WO
UNDS)
7850 IF r>18 THEN  LET STRENGTH=
STRENGTH-49
7860 GO SUB 1320
7870 IF r>0 THEN  GO TO 7920
7880 PRINT "Your defence did not
 protect you."
7890 GO SUB 1520
7900 IF COMBAT=3 THEN  GO TO 803
0
7910 GO TO 7980
7920 PRINT "You defended well."
7930 GO SUB 1520
```

```
7940 FOR r=1 TO 50: NEXT r
7950 IF COMBAT=3 THEN  GO TO 812
0
7960 GO TO 7140
7969 REM **********************
7970 REM MONSTER KILLS YOU
7971 REM **********************
7980 IF WOUNDS>=3 THEN  GO TO 80
30
7990 PRINT "You have been wounde
d."
8000 LET WOUNDS=WOUNDS+1
8010 IF PER>=5 THEN  LET PER=PER
-5
8020 GO TO 7940
8030 PRINT "You have been slain.
"
8040 GO TO 6850
8049 REM **********************
8050 REM COMBAT WITH LICH
8051 REM **********************
8060 LET LICH=INT (RND*200)+500
8070 LET COMBAT=3
8080 LET SP=INT (RND*3)+17
8090 GO SUB 60
8100 PRINT "The Lich attacks. Pr
otect your- self from a ";M$(SP)
8105 LET M(LOCATION)=SP
8110 GO TO 7650
8120 LET LICH=LICH-50
8130 IF LICH>=50 THEN  GO TO 808
0
8140 LET M(LOCATION)=16
8150 LET PER=20
8160 PRINT "The Lich has exhaust
ed its      magic Powers."
8170 PAUSE 200
8180 GO TO 7140
8199 REM **********************
8200 REM LICH'S DEATH
8201 REM **********************
```

```
8210 PRINT "The Lich is dead. Yo
u have won  the magic stone- Eye
 of the StarWarrior. Now you mus
t DESTROY   it. The hand of the
wizard      vanishes. Its job is
 over."
8220 FOR r=1 TO 12
8230 IF H(r)=16 THEN  LET H(r)=2
5
8240 NEXT r
8250 LET WHAND=0: LET STONE=1
8260 GO TO 7490
8489 REM **********************
8490 REM SAVE GAME ROUTINE
8491 REM **********************
8500 SAVE "adventure" LINE 8600
8510 SAVE "graphics" CODE 65368,
166
8520 SAVE "screen"SCREEN$
8530 GO TO 4040
8600 LOAD "graphics"CODE 65368,1
66
8610 LOAD "screen"SCREEN$
8620 BORDER 0: PAPER 0: INK 6
8630 POKE 23658,8
8640 GO TO 4040
8699 REM ***
8700 REM MAP
8701 REM ***
8710 IF FLOOR=1 THEN  LET b=11
8720 IF FLOOR=2 THEN  LET b=111
8730 IF FLOOR=3 THEN  LET b=211
8735 LET f=14: LET g=1
8740 FOR a=b TO 9+b
8750 LET e=0: LET d=6: LET g=g+1
8760 IF O(a)=999 THEN  GO TO 885
0
8770 IF S(a)<>0 THEN  LET d=4
8780 IF M(a)<>0 THEN  LET e=1
8790 LET S$="C"
8800 IF S(a)>0 OR T(a)<>99 THEN
 LET S$="I"
8805 PRINT  INK d; BRIGHT 1; FLA
SH e;AT f,g;S$
```

```
8810 IF a=LOCATION THEN  PRINT A
T f,9;"P"
8820 NEXT a
8830 IF f=21 THEN  GO TO 8860
8835 LET f=f+1: LET 9=1: LET b=b
+10
8840 GO TO 8740
8850 PRINT AT f,9;"1"
8855 GO TO 8820
8860 PRINT AT 17,17;"Press ENTER
"
8865 IF INKEY$="" THEN  GO TO 88
65
8870 GO SUB 60
8873 GO TO 4030
8969 REM **********************
8990 REM SAVE PROGRAM TO AUTO
        RUN
8991 REM **********************
9000 SAVE "Pro9" LINE 5
9010 STOP
```

Table of Variables

a b c d e r	Variables for general use: FOR-NEXT loops etc.
C(X)	C array where x is the value of your location. C array holds the layout of the dungeon. C(x) has the value 999 when it simulates a wall. Otherwise C(x) is given a value representing a room number.
M(x)	Array holds the information needed for the computer to judge what monster if any is in a particular room.
T(x)	Array holds the information needed for the computer to judge what treasure if any is in a particular room. T(x) has the value zero when there is buried treasure and holds the value 99 when there is no treasure in a room.
M$(x)	Holds the names of all the monsters in the game.
T$(x)	Holds the names of all the treasures in the game.
A$(x)	Holds information telling the computer which weapons each of the monsters are vulnerable to.
D$(x)	Holds information telling the computer which defence is useful against each monster.
V(x)	Holds the values for your Combat Rating.
P(x)	Array holds the colour values for each monster.
W(x)	Array holds the colour values for each treasure.
S(x)	Array holds information telling the computer the type and whereabouts of any treasures you might have dropped.
X$(y,x)	Holds the graphic layout of your location. X$ is the graphic display printed top left of your screen display.
SONE	Location of stairs on floor 1.
STWO	Location of stairs on floor 2.
STHREE	Location of stairs on floor 3.
HW	Location of healing well.
FP	Location of fire pit.
FLOOR	Floor level you are on.
LOCATION	Your position in the dungeon.
STRENGTH	The amount of power you have.
WOUNDS	The amount of wounds you have.
HELD	The amount of treasure you have collected. Held can never be greater than 5.
WHAND	Equals zero if you don't have wizard's hand. Equals one if you have wizard's hand.
STONE	Equals zero if you don't have the stone (eye). Equals one if you have the stone.

PER	Your percentage chance of killing the monster at your location. PER obtains its value from array V(x).
SPELL	The number of spells you have collected.
COMBAT	The combat mode you are in. Zero = not in combat. One = You are attacking Two = You are defending. Three = You are defending yourself against the Lich.
PX	Player's X co-ordinate for printing player onto screen.
PY	Player's Y co-ordinate for printing player onto screen. The variables, PY,PX are also used to compare the player's position with the position of objects held in X$(y,x).
SM	Each move the player makes this value is increased by one until SM is greater than eight at which time it is reset to zero and your strength is lowered by a set amount.
WSC	Colour of weapon stored at your location. Obtains its value from W(x).
WFC	Colour of weapon found at your location. Obtains its value from W(x).
MC	Colour of monster at your location. Obtains its value from P(x).
SY	Y co-ordinate of stored treasure at your location.
SX	X co-ordinate of stored treasure at your location. Co-ordinates SY,SX show the screen display position of stored treasures and also their position in X$(y,x).
WY	Y co-ordinate of weapon found at your location.
WX	X co-ordinate of weapon found at your location. Co-ordinates WY,WX show the screen display position of weapons found at your location and also their position in X$(y,x).
BY	Y co-ordinate of buried treasure at your location.
BX	X co-ordinate of buried treasure at your location. Co-ordinates BY,BX show the position of buried treasure on the screen display and in X$(y,x).
MY	Y co-ordinate of monster at your location.
MX	X co-ordinate of monster at your location.
B$(x)	Holds your input to the computer.
k	Marker to tell the computer how many characters are in B$.
COL	Colour of object on the same y,x co-ordinates as player during move player routine.
TRE	Check to see if you have the treasure you wish to use during combat. TRE equals zero if you don't carry that treasure. TRE equals one if you carry the treasure.
WEAPON	The value of the weapon you are using during the combat routine.

TY	Y co-ordinate for testing player's new position during move player routine.
TX	X co-ordinate for testing player's new position during move player routine. TY and TX are also used for testing monster's new position during Monster Moving routine.
RET	Variable for returning the computer to GOSUBS.
RET = 1	then return to GOSUB.
RET = 0	then carry on.

Other titles from Sunshine

THE WORKING SPECTRUM

David Lawrence

0 946408 00 9 £5.95

THE WORKING DRAGON 32

David Lawrence

0 946408 01 7 £5.95

THE WORKING COMMODORE 64

David Lawrence

0 946408 02 5 £5.95

DRAGON 32 GAMES MASTER

Keith Brain/Steven Brain

0 946408 03 03 £5.95

FUNCTIONAL FORTH

for the BBC Computer

Boris Allan

0 946408 04 1 £5.95

COMMODORE 64

machine code master

David Lawrence

0 946408 05 X £6.95

Sunshine also publishes

POPULAR COMPUTING WEEKLY

The first weekly magazine for home computer users. Each copy contains Top 10 charts of the best-selling software and books and up-to-the-minute details of the latest games. Other features in the magazine include regular hardware and software reviews, programming hints, computer swap, adventure corner and pages of listings for the Spectrum, Dragon, BBC, VIC 20 and 64, ZX 81 and other popular micros. Only 35p a week, a year's subscription costs £19.95 (£9.98 for six months) in the UK and £37.40 (£18.70 for six months) overseas.

DRAGON USER

The monthly magazine for all users of Dragon microcomputers. Each issue contains reviews of software and peripherals, programming advice for beginners and advanced users, program listings, a technical advisory service and all the latest news related to the Dragon. A year's subscription (12 issues) costs £8.00 in the UK and £14.00 overseas.

For further information contact:
Sunshine
12–13 Little Newport Street
London WC2R 3LD
01-734- 3454

Printed in England by Commercial Colour Press, London E.7.

NOTES

NOTES

NOTES

Adventuring! You and your trusty computer alone in a dark cave, or deep in the dungeons of a terrifying castle. Alone, that is, except for... What's that? The drip of water on stone, or the approaching footsteps of a minotaur, jaws slavering for your blood?

Spectrum Adventures is in two parts. The first takes a detailed look at the beginnings of Adventure, and how this enthralling pastime has been implemented, in all its guises.

The second part presents a major graphic adventure game called The Eye of the Star Warrior. It was written by Roy Carnell.

Each phase of the game is explained — monster generation, graphics, combat and movement are all described. Many of the subroutines can be used in your own programs.

Tony Bridge is Popular Computing Weekly's regular Adventure columnist. In his other life he sits in a dark underground cavern, making records for most of today's major recording artists.

Roy Carnell founded Carnell Software, a successful company specialising in Adventuring programs. Before that he designed special effects for Star Wars, Superman and many other major films. He is the author of the highly acclaimed Volcanic Dungeon and Black Crystal.

14-03-86 BOGPA £9.95 £5.00

ISBN 0 946408 07 6

£5.95 net